Fundamenta Informaticae XXI (2001) 1–45
IOS Press

Fault Diagnosis with Static and Dynamic Observers*

Franck Cassez†
CNRS/IRCCyN
1 rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France
Email: franck.cassez@cnrs.irccyn.fr

Stavros Tripakis
Cadence Research Laboratories
2150 Shattuck Avenue, 10th floor, Berkeley, CA, 94704, USA
and CNRS, Verimag Laboratory, Centre Equation, 2, avenue de Vignate, 38610 Gières, France
Email: tripakis@cadence.com

Abstract. We study sensor minimization problems in the context of fault diagnosis. Fault diagnosis consists in synthesizing a diagnoser that observes a given plant and identifies faults in the plant as soon as possible after their occurrence. Existing literature on this problem has considered the case of fixed static observers, where the set of observable events is fixed and does not change during execution of the system. In this paper, we consider static observers where the set of observable events is not fixed, but needs to be optimized (e.g., minimized in size). We also consider dynamic observers, where the observer can "switch" sensors on or off, thus dynamically changing the set of events it wishes to observe. It is known that checking diagnosability (i.e., whether a given observer is capable of identifying faults) can be solved in polynomial time for static observers, and we show that the same is true for dynamic ones. On the other hand, minimizing the number of (static) observable events required to achieve diagnosability is NP-complete. We show that this is true also in the case of mask-based observation, where some events are observable but not distinguishable. For dynamic observers' synthesis, we prove that a most permissive finite-state observer can be computed in doubly exponential time, using a game-theoretic approach. We further investigate optimization problems for dynamic observers and define a notion of cost of an observer. We show how to compute an optimal observer using results on mean-payoff games by Zwick and Paterson.

*Preliminary versions of parts of this paper appeared in [2] and [1].
†Work supported by the French Government under grant ANR-SETI-06-003.

1. Introduction

Monitoring, Testing, Fault Diagnosis and Control. Many problems concerning the monitoring, testing, fault diagnosis and control of embedded systems can be formalized using finite automata over a set of *observable* events Σ, plus a set of *unobservable* events [15, 19]. The invisible actions can often be represented by a single unobservable event τ. Given a finite automaton over $\Sigma \cup \{\tau\}$ which is a model of a *plant* (to be monitored, tested, diagnosed or controlled) and an *objective* (good behaviours, what to test for, faulty behaviours, control objective) we want to check if a monitor/tester/diagnoser/controller exists that achieves the objective, and if possible to synthesize one automatically.

The usual assumption in this setting is that the set of observable events is fixed (and this in turn, determines the set of unobservable events as well). Observing an event usually requires some detection mechanism, i.e., a *sensor* of some sort. Which sensors to use, how many of them, and where to place them are some of the design questions that are often difficult to answer, especially without knowing what these sensors are to be used for.

In this paper we study problems of *sensor minimization*. These problems are interesting since observing an event can be costly in terms of time or energy: computation time must be spent to read and process the information provided by the sensor, and power is required to operate the sensor (as well as perform the computations). It is then essential that the sensors used really provide useful information. It is also important for the computer to discard any information given by a sensor that is not really needed. In the case of a fixed set of observable events, it is not the case that all sensors always provide useful information and sometimes energy (used for sensor operation and computer treatment) is spent for nothing. As an example, consider the system described by the automaton $\mathcal{M}$, Fig. 1. $\mathcal{M}$ models a system

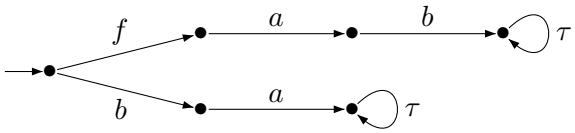


Figure 1. The automaton $\mathcal{M}$.

that has two possible behaviors, a faulty behavior that starts with the fault event f, and a non-faulty behavior that starts with event b. Events a and b are assumed to be observable, whereas events f and τ are unobservable. In this case, a simple way for an observer to detect fault f is to watch for both events a and b: if the sequence $a.b$ is observed, then f must have occurred; if, on the other hand, $b.a$ is observed, then we can be certain that f did not occur (assuming the system under observation behaves precisely as the model $\mathcal{M}$).

Although this is a correct way of performing diagnosis in this case, there exists a "cheaper" way. Initially, the observer turns on only the a sensor (that is, is only able to observe a but not b). When a is observed (eventually it will, since a occurs in both behaviors) the observer turns off the a sensor and turns on the b sensor. If b is observed, then we can conclude that f occurred. As long as b is not observed, f has not occurred.

The former diagnosis method uses a "static" observer in the sense that it watches for the same set of events throughout the entire observation process. The latter method uses a dynamic observer, which

turns sensors on and off as necessary. The latter method can be less costly, as in the example described above.

Sensor Minimization and Fault Diagnosis. We focus our attention on sensor minimization, without looking at problems related to sensor placement, choosing between different types of sensors, and so on. We also focus on a particular observation problem, that of *fault diagnosis*. We believe, however, that the results we obtain are applicable to other contexts as well.

Fault diagnosis consists in observing a plant and detecting whether a fault has occurred or not. We follow the discrete-event system (DES) setting of [16] where the behavior of the plant is assumed to be known: this includes both the nominal, non-faulty behavior of the plant, as well as the faulty behavior of the plant (after a fault occurs). In particular, we assume that we have a model of the plant in the form of a finite-state automaton over $\Sigma \cup \{\tau, f\}$, where Σ is the set of potentially observable events, τ represents the unobservable events, and f is a special unobservable event that corresponds to the faults.[1]

Checking *diagnosability* (whether a fault can be detected) for a given plant and a *fixed* set of observable events can be done in polynomial time [10, 23]. In the general case, synthesizing a diagnoser involves determinization and thus cannot be done in polynomial time.

We examine sensor optimization problems with both *static* and *dynamic* observers. A static observer always observes the same set of events, whereas a dynamic observer can modify the set of events it wishes to observe during the course of the plant execution.

In the static observer case, we consider both the standard setting of observable/unobservable events as well as the setting where the observer is defined as a *mask* which allows some events to be observable but not *distinguishable* (e.g., see [3]). Our first contribution is to show that the problems of *minimizing* the number of observable events (or distinct observable outcomes in case of the mask) are NP-complete. Membership in NP can be easily derived by reducing these problems to the standard diagnosability problem, once a candidate minimal solution is chosen non-deterministically. NP-hardness can be shown using reductions of well-known NP-hard graphs problems, namely the *clique* and *coloring* problems.

In the dynamic observer case, we assume that an observer can decide after each new observation the set of events it is going to watch. We provide a definition of the *dynamic observer synthesis problem* and then show that computing a *dynamic observer* for a given plant, can be reduced to a problem of computing strategies in a *game*. We further investigate optimization problems for dynamic observers and define a notion of *cost* of an observer. Finally we show how to compute an optimal (cost-wise) dynamic observer.

Related work. In Section 3.1 we give a new polynomial time algorithm to check diagnosability. This result itself (testing diagnosability in polynomial time) is not new and was already reported in [10, 23]. Nevertheless the proof we give is original and very simple and applies to plants that are specified by non-deterministic automata with no assumption on non observable loops as in [10, 23]. We can also derive easily a polynomial time algorithm to compute the minimum k such that a DES is k-diagnosable and thus improve a result from [20] but again with a very simple proof.

[1] Different types of faults could also be considered, by having different fault events f_1, f_2, and so on. Our methods can be extended in a straightforward way to deal with multiple faults. We restrict our presentation to a single fault event for the sake of simplicity. Also note that f is assumed to be unobservable. In the case where f can be directly observed, the task of fault diagnosis becomes trivial, since it suffices to report f as soon as it is observed.

NP-hardness of finding minimum-cardinality sets of observable events so that diagnosability holds under the standard, projection-based setting has been previously reported in [22]. Our result of section 3.2 can be viewed as an alternative shorter proof of this result. Masks have not been considered in [22]. As we show in section 4 a reduction from the mask version of the problem to the standard version is not straightforward. Thus the result in section 4 is useful and new.

The complexity of finding "optimal" observation masks, i.e., a set that cannot be reduced, has been considered in [11] where it was shown that the problem is NP-hard for general properties. [11] also shows that finding optimal observation masks is polynomial for "mask-monotonic" properties where increasing the set of observable (or distinguishable) events preserves the property in question. Diagnosability is a mask-monotonic property. However, the notion of "optimality" considered in this work differs from ours: [11] considers the computation of a minimal partition of the set of events w.r.t. partition refinement ordering, and not the computation of the minimum cardinality of the partitions that ensure diagnosability. Notice that optimal observation masks are not the same as minimum-cardinality masks that we consider in this paper.

In [6], the authors investigate the problem of computing a minimal-cost *strategy* that allows to find a subset of the set of observable events such that the system is diagnosable. It is assumed that each such subset has a known associated cost, as well as a known a-priori probability for achieving diagnosability. Dynamic observers are not considered in this work.

Dynamic observers have received little attention in the context of fault diagnosis. The only work we are aware of is the one described in [18]. Our work has been developed independently (Cf. previous versions of this paper [1, 2]). Although the general goals of this work and ours are the same, namely, dynamic on/off switching of sensors in order to achieve cost savings, the setting and the approach of [18] are quite different from ours:

- The definitions of *dynamic observers* differ in the two settings: for instance, for the example of section 4.2.1 of [18], there is no finite-cost solution for Problem CD of [18] (which corresponds to Problem 5 in our paper). In our setting the same example admits a finite-cost solution.

- the purpose of [18] is to compute *one* optimal observation policy (using dynamic programming). In contrast, we solve two problems: first, we compute the set of *all* dynamic observers that can be used to diagnose a system; second we compute all the optimal observers. We do this using well-known game-theoretic approaches.

- there is no upper bound on the complexity of the procedures given in [18]. We show that our solutions are in 2EXPTIME.

- in [18], the authors compute dynamic observers for stochastic systems which we do not consider. They also consider the *control problem under dynamic observation* which we do not consider. Notice that the control problem they consider is a Ramadge & Wonham control problem i.e., on languages on finite words, and thus, the method we propose in paragraph 5.4 could be applied as well.

Organisation of the paper. In Section 2 we fix notation and introduce finite automata with faults to model DES. In Section 3 we present a new algorithm for testing diagnosability. We also show NP-completeness of the sensor minimization problem for the standard projection-based observation setting.

In Section 4 we show NP-completeness of the sensor minimization problem for the mask-based setting. In Section 5 we introduce and study dynamic observers and show that the most permissive dynamic observer can be computed as the most permissive (finite-memory) winning strategy in a safety 2-player game.

We also define a notion of cost for dynamic observers in Section 6 and show that the cost of a given observer can be computed using Karp's algorithm. Finally, we define the optimal-cost observer synthesis problem and show it can be solved using Zwick and Paterson's result on graph games.

2. Preliminaries

2.1. Words and Languages

Let Σ be a finite alphabet and $\Sigma^\tau = \Sigma \cup \{\tau\}$. Σ^* (resp. Σ^ω) is the set of finite (resp. infinite) words over Σ and contains ε which is the *empty* word. We let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. A *language* L is any subset of $\Sigma^* \cup \Sigma^\omega$. Given two words $\rho \in \Sigma^*$ and $\rho' \in \Sigma^* \cup \Sigma^\omega$ we denote $\rho.\rho'$ the concatenation of ρ and ρ' which is defined in the usual way. $|\rho|$ stands for the length of the word ρ (the length of the empty word is zero, the length of an infinite word is ∞) and $|\rho|_\lambda$ with $\lambda \in \Sigma$ stands for the number of occurrences of λ in ρ. We also use the notation $|S|$ to denote the cardinality of a set S. Given $\Sigma_1 \subseteq \Sigma$, we define the *projection* operator on finite words, $\boldsymbol{\pi}_{/\Sigma_1} : \Sigma^* \to \Sigma_1^*$, recursively as follows: $\boldsymbol{\pi}_{/\Sigma_1}(\varepsilon) = \varepsilon$ and for $a \in \Sigma, \rho \in \Sigma^*$, $\boldsymbol{\pi}_{/\Sigma_1}(a.\rho) = a.\boldsymbol{\pi}_{/\Sigma_1}(\rho)$ if $a \in \Sigma_1$ and $\boldsymbol{\pi}_{/\Sigma_1}(\rho)$ otherwise.

2.2. Automata

Definition 2.1. (Finite Automaton)
An *automaton* A is a tuple $(Q, q_0, \Sigma^\tau, \delta, F, R)$ with Q a set of states, $q_0 \in Q$ is the initial state, $\delta \subseteq Q \times \Sigma^\tau \times 2^Q$ is the transition relation and $F \subseteq Q$, $R \subseteq Q$ are receptively the set of final and repeated states. We write $q \xrightarrow{\lambda} q'$ if $q' \in \delta(q, \lambda)$. For $q \in Q$, *Enabled*(q) is the set of actions *enabled* at q, i.e., the set of λ such that $q \xrightarrow{\lambda} q'$ for some q'.
If Q is finite, A is a *finite automaton*. An automaton is *deterministic* if for any $q \in Q$, $|\delta(q, \tau)| = 0$ and for any $\lambda \in \Sigma$, $|\delta(q, \lambda)| \leq 1$. A *labeled* automaton is a pair (A, L) where $A = (Q, q_0, \Sigma^\tau, \delta, F, R)$ is an automaton and $L : Q \to P$ where P is a finite set of *observations*. ■

A *run* ρ from state s in A is a finite or infinite sequence of transitions

$$s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \cdots s_{n-1} \xrightarrow{\lambda_n} s_n \cdots \tag{1}$$

s.t. $\lambda_i \in \Sigma^\tau$ and $s_0 = s$. If ρ is finite, of length n, and ends in s_n we let *tgt*$(\rho) = s_n$. The set of finite (resp. infinite) runs from s in A is denoted *Runs*(s, A) (resp. *Runs*$^\omega(s, A)$) and we define *Runs*(A) = *Runs*(q_0, A) and *Runs*$^\omega(A)$ = *Runs*$^\omega(q_0, A)$. The *trace* of the run ρ, denoted *tr*(ρ), is the word obtained by concatenating the symbols λ_i appearing in ρ, for those λ_i different from τ. The finite word $w \in \Sigma^*$ is *accepted* by A if $w =$ *tr*(ρ) for some $\rho \in$ *Runs*(A) with *tgt*$(\rho) \in F$. The *language of finite words* $\mathcal{L}(A)$ of A is the set of words accepted by A. The infinite word $w \in \Sigma^\omega$ is accepted by A if $w =$ *tr*(ρ) for some $\rho \in$ *Runs*$^\omega(A)$ and there is an infinite set of indices $I = \{i_0, i_1, \cdots, i_k, \cdots\}$ s.t. $s_l \in R$ for each $l \in I$. The *language of infinite words* $\mathcal{L}^\omega(A)$ of A is the set of infinite words accepted by A.

2.3. Discrete-Event Systems

In this paper we use finite automata that generate prefix-closed languages of finite words to model discrete-event systems, hence we do not always need a set of final or repeated states (we assume $F = Q$ and $R = \emptyset$ if not otherwise stated).

Let $f \notin \Sigma^\tau$ be a fresh letter that corresponds to the fault action. We also let $\Sigma^{\tau,f} = \Sigma^\tau \cup \{f\}$ and $A = (Q, q_0, \Sigma^{\tau,f}, \delta)$. Given $V \subseteq \mathit{Runs}(A)$, $\mathit{Tr}(V) = \{\mathit{tr}(\rho) \text{ for } \rho \in V\}$ is the set of traces of the runs in V. A run ρ as in equation (1) is *k-faulty* if there is some $1 \leq i \leq n$ s.t. $\lambda_i = f$ and $n - i \geq k$. Notice that ρ can be infinite and in this case $n = \infty$ and $n - i \geq k$ always holds. $\mathit{Faulty}_{\geq k}(A)$ is the set of k-faulty runs of A. A run is *faulty* if it is k-faulty for some $k \in \mathbb{N}$ and $\mathit{Faulty}(A)$ denotes the set of faulty runs. It follows that $\mathit{Faulty}_{\geq k+1}(A) \subseteq \mathit{Faulty}_{\geq k}(A) \subseteq \cdots \subseteq \mathit{Faulty}_{\geq 0}(A) = \mathit{Faulty}(A)$. Finally, $\mathit{NonFaulty}(A) = \mathit{Runs}(A) \setminus \mathit{Faulty}(A)$ is the set on *non-faulty* runs of A. We let $\mathit{Faulty}^{tr}_{\geq k}(A) = \mathit{Tr}(\mathit{Faulty}_{\geq k}(A))$ and $\mathit{NonFaulty}^{tr}(A) = \mathit{Tr}(\mathit{NonFaulty}(A))$ be the sets of traces of faulty and non-faulty runs.

We assume that each faulty run of A of length n can be extended into a run of length $n + 1$. This is required for technical reasons (in order to guarantee that the set of faulty runs where sufficient "logical" time has elapsed after the fault is well-defined) and can be achieved by adding τ loop-transitions to each deadlock state of A. Notice that this transformation does not change the observations produced by the plant, thus, any observer synthesized for the transformed plant also applies to the original one.

2.4. Product of Automata

The product of automata with τ-transitions is defined in the usual way: the automata synchronize on common labels except for τ. Let $A_1 = (Q_1, q_0^1, \Sigma_1^\tau, \delta_1)$ and $A_2 = (Q_2, q_0^2, \Sigma_2^\tau, \delta_2)$. The *product* of A_1 and A_2 is the automaton $A_1 \times A_2 = (Q, q_0, \Sigma^\tau, \delta)$ where:

- $Q = Q_1 \times Q_2$,
- $q_0 = (q_0^1, q_0^2)$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\delta \subseteq Q \times \Sigma^\tau \times Q$ is defined by $(q_1, q_2) \xrightarrow{\sigma} (q_1', q_2')$ if:
  - either $\sigma \in \Sigma_1 \cap \Sigma_2$ and $q_k \xrightarrow{\sigma}_k q_k'$, for $k = 1, 2$,
  - or $\sigma \in (\Sigma_i \setminus \Sigma_{3-i}) \cup \{\tau\}$ and $q_i \xrightarrow{\sigma}_i q_i'$ and $q_{3-i}' = q_{3-i}$, for $i = 1$ or $i = 2$.

3. Sensor Minimization with Static Observers

In this section we address the sensor minimization problem for *static observers*. We point out that the result in this section was already obtained in [21] and we only give here an alternative shorter proof. We are given a finite automaton $A = (Q, q_0, \Sigma^{\tau,f}, \delta)$. Such an automaton A models a *plant*. Notice that A is as in Definition 2.1 with the addition that its alphabet includes both the unobservable non-fault event τ and the unobservable fault event f.

A *diagnoser* is a device that observes the plant and raises an "alarm" whenever it detects a fault. We allow the diagnoser to raise an alarm not necessarily immediately after the fault occurs, but possibly some time later, as long as this time is bounded by some $k \in \mathbb{N}$, where $\mathbb{N}$ is the set of non-negative integers. We model time by counting the "moves" the plant makes (including observable and unobservable ones). If the system generates a word ρ but only a subset $\Sigma_o \subseteq \Sigma$ is observable, the diagnoser can only see $\boldsymbol{\pi}_{/\Sigma_o}(\rho)$.

Definition 3.1. ((Σ_o, k)-Diagnoser)
Let A be a finite automaton over $\Sigma^{\tau,f}$, $k \in \mathbb{N}$, $\Sigma_o \subseteq \Sigma$. A mapping $D : \Sigma_o^* \to \{0,1\}$ is a (Σ_o, k)-*diagnoser for* A if:

- for each $\rho \in \mathit{NonFaulty}(A)$, $D(\boldsymbol{\pi}_{/\Sigma_o}(\mathit{tr}(\rho))) = 0$,
- for each $\rho \in \mathit{Faulty}_{\geq k}(A)$, $D(\boldsymbol{\pi}_{/\Sigma_o}(\mathit{tr}(\rho))) = 1$. ■

A is (Σ_o, k)-diagnosable if there is a (Σ_o, k)-diagnoser for A. A is Σ_o-diagnosable if there is some $k \in \mathbb{N}$ s.t. A is (Σ_o, k)-diagnosable.

Remark 3.1. At this stage, we do not require the mapping D to be effectively computable or even regular.

Example 3.1. Let $\mathcal{A}$ be the automaton shown on Fig. 2. The run with one action f is in $\mathit{Faulty}_{\geq 0}(\mathcal{A})$, the run $f.a$ is in $\mathit{Faulty}_{\geq 1}(\mathcal{A})$ and $a.\tau^2$ is in $\mathit{NonFaulty}(\mathcal{A})$. $\mathcal{A}$ is neither $\{a\}$-diagnosable, nor $\{b\}$-

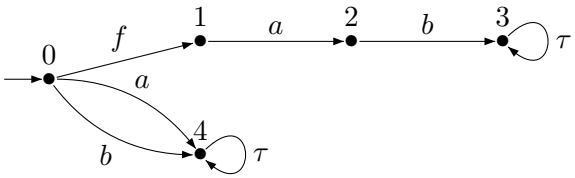


Figure 2. The automaton $\mathcal{A}$.

diagnosable. This is because, for any k, the faulty run $f.a.b.\tau^k$ gives the same observation as the non-faulty run $a.\tau^k$ (in case a is the observable event) or the non-faulty run $b.\tau^k$ (in case b is the observable event). Consequently, the diagnoser cannot distinguish between the two no matter how long it waits. If both a and b are observable, however, we can define: $D(a.b.\rho) = 1$ for any $\rho \in \{a,b\}^*$ and $D(\rho) = 0$ otherwise. D is a $(\{a,b\}, 2)$-diagnoser for $\mathcal{A}$, thus, $\mathcal{A}$ is $\{a,b\}$-diagnosable.

3.1. Checking Diagnosability

Let $A = (Q, q_0, \Sigma^{\tau,f}, \to)$ be a finite automaton over $\Sigma^{\tau,f}$. In this subsection we give a polynomial time algorithm to check diagnosability.

As reported in [10, 23], checking whether A is Σ-diagnosable can be done in polynomial time in the size of A, more precisely in $O(|A|^2)$. We improve the previous results of [10, 23, 20] on this subject with respect to the following directions:

- compared to [23], the product automaton (denoted $A_1 \times A_2$ in the sequel) that we build has fewer states; moreover our result applies even if the plant is given by a non-deterministic automaton with non-observable loops.
- in [10], in case there are many types of faults, the algorithm for checking diagnosability runs in exponential time. We can use our algorithm to check each fault at a time and obtain a polynomial-time algorithm for multiple types of faults.
- to compute the maximum delay k to diagnose a system, [20] computes the strongly connected components of a product automaton and then reduces the problem of computing k, to a shortest-path problem in a graph. This proves that k can be computed in cubic time in the size of the plant. We improve this result with our construction and obtain a quadratic algorithm.

Altogether, this section provides a uniform and elegant setting to check diagnosability of discrete-event systems with multiple faults types and to compute the maximum delay. Moreover the proofs are very easy and short. It also reduces the diagnosability problem to the Büchi emptiness problem: very efficient algorithms [4] are known to solve this problem, for instance *on-the-fly* algorithms [8]. In this respect, our algorithm can be very useful because it is implemented in very efficient tools like SPIN [9] to check for Büchi emptiness.

An Algorithm to Check Diagnosability. Let $A = (Q, q_0, \Sigma^{\tau,f}, \rightarrow)$ be a finite automaton over $\Sigma^{\tau,f}$. We assume that $\Sigma_o = \Sigma$ i.e., all the events in Σ are observable and thus τ is the only unobservable action. By definition 3.1, A is diagnosable iff $\exists k \in \mathbb{N}^*$ s.t. A is (Σ, k)-diagnosable. This is equivalent to:

$$A \text{ is not diagnosable} \iff \forall k \in \mathbb{N}^*, A \text{ is not } (\Sigma, k)\text{-diagnosable} \tag{2}$$

$$\iff \forall k \in \mathbb{N}^*, \begin{cases} \exists \rho \in \mathit{NonFaulty}(A) \text{ and } \exists \rho' \in \mathit{Faulty}_{\geq k}(A) \\ \text{s.t. } \boldsymbol{\pi}_{/\Sigma}(tr(\rho)) = \boldsymbol{\pi}_{/\Sigma}(tr(\rho')) \end{cases} \tag{3}$$

There is also a language-based definition of (Σ, k)-diagnosability: A is (Σ, k)-diagnosable iff

$$\boldsymbol{\pi}_{/\Sigma}(\mathit{Faulty}^{tr}_{\geq k}(A)) \cap \boldsymbol{\pi}_{/\Sigma}(\mathit{NonFaulty}^{tr}(A)) = \emptyset \tag{4}$$

or in other words, there is no pair of runs (ρ_1, ρ_2) with $\rho_1 \in \mathit{Faulty}_{\geq k}(A)$, $\rho_2 \in \mathit{NonFaulty}(A)$ s.t. ρ_1 and ρ_2 give the same observations on Σ. To check diagnosability, we build a product automaton $A_1 \times A_2$ such that A_1 behaves like A but records whether a fault occurred or not, and A_2 produces only the non-faulty runs of A. Define $A_1 = (Q \times \{0,1\}, (q_0, 0), \Sigma^\tau, \rightarrow_1)$ s.t.

- $(q, n) \xrightarrow{l}_1 (q', n)$ iff $q \xrightarrow{l} q'$ with $l \in \Sigma$;
- $(q, n) \xrightarrow{\tau}_1 (q', 1)$ iff $q \xrightarrow{f} q'$, (n is set to 1 after a fault occurs);
- $(q, n) \xrightarrow{\tau}_1 (q', n)$ iff $q \xrightarrow{\tau} q'$.

Define $A_2 = (Q, q_0, \Sigma^\tau, \rightarrow_2)$ with

- $q \xrightarrow{l}_2 q'$ if $q \xrightarrow{l} q'$ and $l \in \Sigma$;

- $q \xrightarrow{\tau}_2 q'$ if $q \xrightarrow{\tau} q'$.

Let $A_1 \times A_2$ be the synchronized product of A_1 and A_2: synchronization happens on common actions except τ, i.e., actions in Σ and otherwise interleaving (Cf. section 2.4). $\rightarrow_{1,2}$ stands for the transition relation of $A_1 \times A_2$. We assume we have a predicate A_1*Move* (resp. A_2*Move*) on transitions of $A_1 \times A_2$ which is true when A_1 (resp. A_2) participates in an $A_1 \times A_2$ transition (i.e., A_1*Move*(t) is true for all transitions except τ transitions taken by A_2). For a run $\rho \in$ *Runs*$(A_1 \times A_2)$ we let $\rho_{|1}$, be the run of A_1 defined by the sequence of A_1*Move* transitions of ρ and $\rho_{|2}$ be the sequence of A_2*Move* moves. Notice that $\boldsymbol{\pi}_{/\Sigma}(tr(\rho_{|1})) = \boldsymbol{\pi}_{/\Sigma}(tr(\rho_{|2}))$ by definition of $A_1 \times A_2$. Let *Runs*$_{\geq k}(A_1 \times A_2)$ be the set of runs in $A_1 \times A_2$ s.t. a (faulty) state $((q_1, 1), q_2)$ is followed by at least k A_1-actions. Then we have the following lemmas:

Lemma 3.1. Let $\rho \in$ *Runs*$_{\geq k}(A_1 \times A_2)$. Then $\rho_{|1} \in$ *Faulty*$_{\geq k}(A)$ and $\rho_{|2} \in$ *NonFaulty*(A). Moreover $\boldsymbol{\pi}_{/\Sigma}(tr(\rho_{|1})) = \boldsymbol{\pi}_{/\Sigma}(tr(\rho_{|2}))$.

Lemma 3.2. Let $\rho_1 \in$ *Faulty*$_{\geq k}(A)$ and $\rho_2 \in$ *NonFaulty*(A) s.t. $\boldsymbol{\pi}_{/\Sigma}(tr(\rho_1)) = \boldsymbol{\pi}_{/\Sigma}(tr(\rho_2))$. Then there is some $\rho \in$ *Runs*$_{\geq k}(A_1 \times A_2)$ s.t. $\rho_{|1} = \rho_1$ and $\rho_{|2} = \rho_2$.

Proof:
By definition of $A_1 \times A_2$, it suffices to notice that a run in *Runs*$_{\geq k}(A_1 \times A_2)$ can be split into a run of A_1 and a run of A_2 having the same projection on Σ. □

We can define an automaton $\mathcal{B}$ which is an extended version of $A_1 \times A_2$: we add a boolean variable z that is set to 0 in the initial state of $\mathcal{B}$. An extended state of $\mathcal{B}$ is a pair (s, z) with s a state of $A_1 \times A_2$. Whenever A_1 participates in an $A_1 \times A_2$-action, z is set to 1, and when only A_2 makes a move in $A_1 \times A_2$, z is set 0. We denote $\longrightarrow_{\mathcal{B}}$ the new transition relation between extended states; $(s, z) \xrightarrow{\sigma}_{\mathcal{B}} (s', z')$ if there is a transition $t : s \xrightarrow{\sigma}_{1,2} s'$, and $z' = 1$ if A_1*Move*(t), and $z' = 0$ otherwise. Define the set of states $R_{\mathcal{B}} = \{(((q, 1), q'), 1) \,|\, ((q, 1), q') \in A_1 \times A_2\}$. The Büchi automaton $\mathcal{B}$ is the tuple $((Q \times \{0, 1\} \times Q) \times \{0, 1\}, ((q_0, 0), q_0, 0), \Sigma^\tau, \longrightarrow_{\mathcal{B}}, \emptyset, R_{\mathcal{B}})$ with $R_{\mathcal{B}}$ the set of repeated states. The following theorem holds:

Theorem 3.1. $\mathcal{L}^\omega(\mathcal{B}) \neq \emptyset \iff A$ *is not* Σ*-diagnosable.*

Proof:
$\boxed{\Longrightarrow}$ Assume $\mathcal{L}^\omega(\mathcal{B}) \neq \emptyset$. Let $\rho \in \mathcal{L}^\omega(\mathcal{B})$: ρ has infinitely many faulty states and infinitely many A_1-actions because of the definition of $R_{\mathcal{B}}$. Let $k \in \mathbb{N}$. Let $\rho[i_k]$ be a finite prefix of ρ that contains more than k A_1-actions (for any k this i_k exists because ρ contains infinitely many A_1-actions). $\rho[i_k] \in$ *Runs*$_{\geq k}(A_1 \times A_2)$ and by Lemma 3.1, it follows that $\rho[i_k]_{|1} \in$ *Faulty*$_{\geq k}(A)$ and $\rho[i_k]_{|2} \in$ *NonFaulty*(A) and $\boldsymbol{\pi}_{/\Sigma}(tr(\rho_{|1})) = \boldsymbol{\pi}_{/\Sigma}(tr(\rho_{|2}))$. Thus equation (3) is satisfied and A is not diagnosable.
$\boxed{\Longleftarrow}$ Conversely assume A is not Σ-diagnosable. Then by equation (3) and Lemma 3.2, for any $k \in \mathbb{N}$, there is a run $\rho_k \in$ *Runs*$_{\geq k}(A_1 \times A_2)$. As $A_1 \times A_2$ is finite, there must be a state $(q_1, 1), q_2$ in $A_1 \times A_2$ which is the source of an infinite number of runs ρ'_k having more than k A_1-actions. As $A_1 \times A_2$ is of finite branching, by König's Lemma, there is a infinite run in $A_1 \times A_2$ containing infinitely many A_1-actions and thus $\mathcal{L}^\omega(\mathcal{B}) \neq \emptyset$. □

This gives a procedure to check whether an automaton A is diagnosable or not. The product (extended) system built from $A_1 \times A_2$ has size $4 \cdot |A|^2$ i.e., $O(|A|^2)$. Checking emptiness of a Büchi automaton having n states and m transitions can be done in $O(n+m)$. Thus, checking diagnosability of a finite (non-deterministic) automaton can be done in $O(|A|^2)$ i.e., is also polynomial.

In the case of a finite number of *failure types*, it suffices to design $A_1 \times A_2$ for each type of faults [23] and check for diagnosability. Thus checking diagnosability with multiple type of faults is also polynomial.

Computation of the Maximal Delay to Detect Faults. To compute the smallest k s.t. A is (Σ, k)-diagnosable we proceed as follows: in case A is Σ-diagnosable, there is no infinite faulty run with infinitely many A_1-actions in $A_1 \times A_2$; this implies that each run beginning in a faulty state of $A_1 \times A_2$ is followed by a finite number of A_1-actions and this number is bounded by a constant α. Indeed, otherwise, we could construct an infinite faulty run with a infinite number of A_1-actions and A would not be diagnosable. Assume k is the maximum number of A_1-actions that can follow a faulty state in $A_1 \times A_2$. Then A is $(\Sigma, k+1)$-diagnosable: otherwise there would be a faulty run with $k+1$ A_1-actions after a faulty state in $A_1 \times A_2$. Also A is not (Σ, k)-diagnosable because there is faulty run with k A_1-actions after a faulty state in $A_1 \times A_2$, which also means there is another non faulty run with the same observable actions. Thus $k+1$ is the minimum value for which A is Σ-diagnosable or in other words, it is the maximal delay after which a fault will be announced. Computing $k+1$ amounts to computing the maximum number of A_1-actions that can follow a faulty state in $A_1 \times A_2$.

The computation of k amounts to computing the length of a longest path[2] in a graph which can be done in linear time w.r.t. size of the graph. As the size of $A_1 \times A_2$ is $O(|Q|^2)$ we can compute k in quadratic time which is better than the bound $O(|Q|^3)$ of [20].

Synthesis of a Diagnoser. To compute the diagnoser we do the following: take A_1 and determinize it (using the standard subset construction). The deterministic automaton A' obtained this way has at most $2^{O(|Q|)}$ states. If a state S of the deterministic version A' contains only faulty states, i.e., every $(q,k) \in S$ is such that $k = 1$ then declare S as faulty.

Define the diagnoser D as follows: for a run $\rho \in \mathit{Runs}(A)$, $D(\rho) = 1$ if $S_0 \xrightarrow{tr(\rho)} S'$ in A' and S' is faulty; otherwise $D(\rho) = 0$. D is a diagnoser for A and the maximum delay to announce a fault is $k+1$ steps if k is the minimum value for which A is (Σ, k)-diagnosable. Computing the diagnoser is thus exponential in the size of A, i.e., can be done in $2^{O(|A|)}$.

3.2. Minimum Cardinality Set for Static Observers

In this section we address the problem of *finding* a set of observable events Σ_o that allows faults to be detected. We would like to detect faults using as few observable events as possible. We want to decide whether there is a subset $\Sigma_o \subsetneq \Sigma$ such that the fault can be detected by observing only events in Σ_o. Moreover, we would like to find an "optimal" such Σ_o (e.g., one with minimal number of events).

[2] Actually it amounts to computing the maximum number of A_1-actions following a faulty state in $A_1 \times A_2$. This can be done using a depth-first search algorithm and a tag on each faulty state s that gives the maximal number of A_1-actions following s.

Problem 1. (Minimum Number of Observable Events)

INPUT: Plant model $A = (Q, q_0, \Sigma^{\tau,f}, \delta)$, $n \in \mathbb{N}$ s.t. $n \leq |\Sigma|$.
PROBLEMS:

(A) Is there any $\Sigma_o \subseteq \Sigma$ with $|\Sigma_o| = n$, such that A is Σ_o-diagnosable ?

(B) If the answer to (A) is "yes", find the minimum n_0 such that there exists $\Sigma_o \subseteq \Sigma$ with $|\Sigma_o| = n_0$ and A is Σ_o-diagnosable.

If we know how to solve Problem 1(A) efficiently then we can also solve Problem 1(B) efficiently: we perform a binary search over n between 0 and $|\Sigma|$, and solve Problem 1(A) for each such n, until we find the minimum n_0 for which Problem 1(A) gives a positive answer.[3] Unfortunately, Problem 1(A) is a combinatorial problem, exponential in $|\Sigma|$, as we show next.

Theorem 3.2. Problem 1(A) is NP-complete.

Proof:
Membership in NP is proved using the result in section 3.1: if we guess a solution Σ_o we can check that A is Σ_o-diagnosable in time polynomial in $|A|$. Here we provide the proof of NP-hardness by giving a reduction of the n-clique problem. Let $G = (V, E)$ be a (undirected) graph where V is set of vertices and $E \subseteq V \times V$ is a set of edges (we assume that $(v, v') \in E \iff (v', v) \in E$). A *clique* in G is a subset $V' \subseteq V$ such that for all $v, v' \in V'$, $(v, v') \in E$. The *n-clique* problem asks the following: determine whether G contains a clique of n vertices.

The reduction is as follows. Given G, we build a finite automaton A_G such that there is a n-clique in G iff A_G is Σ_o-diagnosable, where $|\Sigma \setminus \Sigma_o| = n$. Notice that since $\Sigma_o \subseteq \Sigma$, $|\Sigma \setminus \Sigma_o| = n$ is equivalent to $|\Sigma_o| = |\Sigma| - n$. Thus, there is a n-clique in G iff n events from Σ do not need to be observed i.e., iff Problem 1(A) gives a positive answer for $|\Sigma| - n$.

Starting from G we define $\Sigma = V$. Then we define A_G as shown in Fig. 3: q_0 is the initial state and the "branches" $f.a.b$, $f.a'.b'$, $\cdots$ are obtained from the pairs of nodes $(a, b), (a', b'), \cdots$ that are *not* linked with an edge in G: $(a, b), (a', b'), \cdots \notin E$.

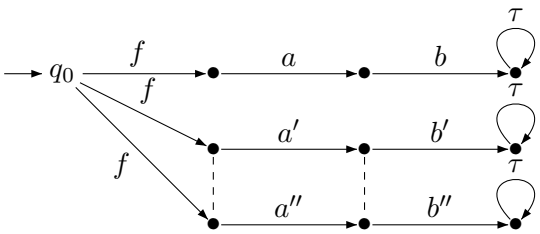


Figure 3. The automaton A_G.

[3] Notice that knowing n_0 does not imply we know the required set of observable events Σ_o. We can find (one of the possibly many) Σ_o by searching over all possible subsets $\Sigma_o \subseteq \Sigma$ of size n_0 (there are $C(|\Sigma|, n_0)$ such combinations) and check for each such Σ_o whether A is Σ_o-diagnosable, using the methods described in section 3.1.

If part. Assume Problem 1(A) gives a positive answer for $|\Sigma|-n$. This means that there exists $\Sigma_o \subseteq \Sigma$ such that $|\Sigma_o| = |\Sigma| - n$, or $|\Sigma \setminus \Sigma_o| = n$, and A_G is Σ_o-diagnosable. Then we claim that $C = \Sigma \setminus \Sigma_o$ is a clique in G. Assume C is not a clique. Then there must be some $\{a, b\} \subseteq C$ such that $(a, b) \notin E$. By the construction of Fig. 3, there is a branch $f.a.b$ in A_G and thus both $\rho = \varepsilon$ and $\rho' = f.a.b.\tau^k$ are runs of A_G, for any k. As $\{a, b\} \subseteq C$, $\{a, b\} \cap \Sigma_o = \emptyset$ and the observation of both runs[4] ρ and ρ' is ε, no diagnoser exists that can distinguish between the two runs, for any k.

Only if part. Assume there exists a n-clique in G. Let $\Sigma_o = \Sigma \setminus C$. We claim that A_G is $(\Sigma_o, 2)$-diagnosable (thus also Σ_o-diagnosable). Suppose not. Then there exist two runs ρ and ρ' such that $\rho' \in \mathit{NonFaulty}(A_G)$ and $\rho = \rho_1.f.\rho_2$, and $|\rho_2| \geq 2$, and $\boldsymbol{\pi}_{/\Sigma_o}(\rho) = \boldsymbol{\pi}_{/\Sigma_o}(\rho')$. Then, ρ must be of the form $\rho = f.a.b.\tau^k$, with $(a, b) \notin E$. Also, the only way ρ' can be non faulty is $\rho' = \varepsilon$. Then $\boldsymbol{\pi}_{/\Sigma_o}(\rho') = \varepsilon = \boldsymbol{\pi}_{/\Sigma_o}(\rho)$, thus, both a and b must be non-observable, thus $\{a, b\} \cap \Sigma_o = \emptyset$ which entails $\{a, b\} \subseteq C$. This implies that $(a, b) \in E$ which is a contradiction. □

Remark 3.2. Notice that in Problem 1, the maximum number of steps after which a fault has to be reported, k, is not specified as an input. Adding k to the set of inputs results in answers that may generally depend on k. For instance, in the automaton $\mathcal{E}$ of Fig. 4, if we impose that a fault be reported within $k = 2$ steps, then we need to observe both a and b; whereas if we leave k unspecified, we need only observe a and we can diagnose a fault after 3 steps (i.e., 2 a's).

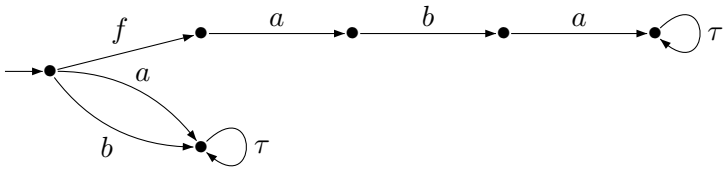


Figure 4. The automaton $\mathcal{E}$.

4. Sensor Minimization with Masks

So far we have assumed that observable events are also *distinguishable*. However, there are cases where two events a and b are observable but not distinguishable, that is, the diagnoser knows that a or b occurred, but not which of the two. This is not the same as considering a and b to be unobservable, since in that case the diagnoser would not be able to detect occurrence of a or b. Distinguishability of events is captured by the notion of a *mask* [3].

Definition 4.1. (Mask)
A *mask* (M, n) over Σ is a total, surjective function $M : \Sigma \to \{1, \cdots, n\} \cup \{\varepsilon\}$. ■

M induces a morphism $M^* : \Sigma^* \to \{1, \cdots, n\}^*$, where $M^*(\varepsilon) = \varepsilon$ and $M^*(a.\rho) = M(a).M^*(\rho)$, for $a \in \Sigma$ and $\rho \in \Sigma^*$. For example, if $\Sigma = \{a, b, c, d\}$, $n = 2$ and $M(a) = M(b) = 1$, $M(c) = 2$, $M(d) = \varepsilon$, then we have $M^*(a.b.c.b.d) = 1.1.2.1 = M^*(a.a.d.c.a)$.

[4] When a run can be uniquely defined by its sequence of actions, we omit the intermediate states and we can use the trace to characterize the run.

Definition 4.2. ($(M,n),k$)-diagnoser)
Let (M,n) be a mask over Σ. A mapping $D : \{1,\cdots,n\}^* \to \{0,1\}$ is a $((M,n),k)$-*diagnoser* for A if:

- for each $\rho \in \mathit{NonFaulty}(A)$, $D(M^*(\boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho)))) = 0$;
- for each $\rho \in \mathit{Faulty}_{\geq k}(A)$, $D(M^*(\boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho)))) = 1$. ■

A is $((M,n),k)$-diagnosable if there is a $((M,n),k)$-diagnoser for A. A is said to be (M,n)-diagnosable if there is some k such that A is $((M,n),k)$-diagnosable.

Given A and a mask (M,n), checking whether A is $((M,n)$-diagnosable can be done in polynomial time. In fact it can be reduced to checking Σ_o-diagnosability of a modified automaton A_M, with $\Sigma_o = \{1,...,n\}$. A_M is obtained from A by renaming the actions $a \in \Sigma$ by $M(a)$ or τ if $M(a) = \varepsilon$. It can be seen that A is $((M,n)$-diagnosable iff A_M is $\{1,\cdots,n\}$-diagnosable. Also notice that A is $((M,n),k)$-diagnosable iff

$$M^*(\boldsymbol{\pi}_{/\Sigma}(\mathit{Faulty}^{tr}_{\geq k}(A))) \cap M^*(\boldsymbol{\pi}_{/\Sigma}(\mathit{NonFaulty}^{tr}(A))) = \emptyset.$$

As in the previous section, we are mostly interested in minimizing the observability requirements while maintaining diagnosability. In the context of diagnosis with masks, this means minimizing the number n of distinct outputs of the mask M. We thus define the following problem:

Problem 2. (Minimum Mask)

INPUT: Plant model $A = (Q, q_0, \Sigma^{\tau,f}, \delta)$, $n \in \mathbb{N}$ s.t. $n \leq |\Sigma|$.
PROBLEM:

(A) Is there any mask (M,n) such that A is (M,n)-diagnosable ?

(B) If the answer to (A) is "yes", find the minimum n_0 such that there is a mask (M,n_0) such that A is (M,n_0)-diagnosable.

As with Problem 1, if we know how to solve Problem 2(A) efficiently we also know how to solve Problem 2(B) efficiently: again, a binary search on n suffices.

We will prove that Problem 2 is NP-complete. One might think that this result follows easily from Theorem 3.2. However, this is not the case. Obviously, a solution to Problem 1 provides a solution to Problem 2: assume there exists Σ_o such that A is (Σ_o,k)-diagnosable and $\Sigma_o = \{a_1,...,a_n\}$; define a mask $M : \Sigma \to \{1,\cdots,n\}$ such that $M(a_i) = i$ and for any $a \in \Sigma \setminus \Sigma_o$, $M(a) = \varepsilon$. Then, A is $((M,n),k)$-diagnosable. However, a positive answer to Problem 2(A) does not necessarily imply a positive answer to Problem 1(A), as shown by the example that follows.

Example 4.1. Consider again the automaton $\mathcal{A}$ of Fig. 2. Let $M(a) = M(b) = 1$. Then $\mathcal{A}$ is $((M,1),2)$-diagnosable because we can build a diagnoser D defined by: $D(\varepsilon) = 0, D(1) = 0, D(1^2.\rho) = 1$ for any $\rho \in 1^*$. However, as we said before, there is no strict subset of $\{a,b\}$ that allows A to be diagnosed.[5]

[5] Note that automaton $\mathcal{B}$ of Fig. 6, although looking very similar to $\mathcal{A}$, is not $(M,1)$-diagnosable.

Instead of using the previous result, we provide a direct proof of NP-hardness of Problem 2.

Theorem 4.1. Problem 2 is NP-complete.

Proof:
Membership in NP is again justified by the fact that checking whether a guessed mask works can be done in polynomial time (it suffices to rename the events of the system according to M and apply the algorithm of section 3.1). We show NP-hardness using a reduction of the n-coloring problem. The n-coloring problem asks the following: given an undirected graph $G = (V, E)$, is it possible to color the vertices with colors in $\{1, 2, \cdots, n\}$ so that no two adjacent vertices have the same color ?

Let $G = (V, E)$ be an undirected graph. Let $E = \{e_1, e_2, \cdots, e_j\}$ be the set of edges with $e_i = (u_i, v_i)$. We let $\Sigma = V$ and define the automaton A_G as pictured in Fig. 5. The initial state of A_G is q_0.

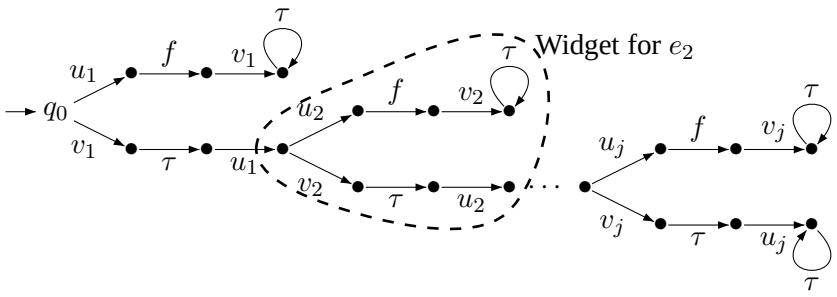


Figure 5. Automaton A_G for n-colorizability.

We claim that G is n-colorizable iff A_G is (M, n)-diagnosable.

If part Assume A_G is (M, n)-diagnosable for $n \geq 0$. We first show that for all $i = 1, ..., j$, $M(u_i) \neq \varepsilon$, $M(v_i) \neq \varepsilon$ and $M(u_i) \neq M(v_i)$. For any k, we can define the runs $\rho = v_1.\tau.u_1.v_2.\tau.u_2 \cdots u_i.f.v_i.\tau^k$ and $\rho' = v_1.\tau.u_1.v_2.\tau.u_2 \cdots v_i.\tau.u_i$. If either $M(u_i) = \varepsilon$ or $M(v_i) = \varepsilon$ or $M(u_i) = M(v_i)$ holds, then we have $M^*(\boldsymbol{\pi}_{/\Sigma}(tr(\rho))) = M^*(\boldsymbol{\pi}_{/\Sigma}(tr(\rho')))$. Thus, for any k, there is a faulty run with more than k events after the fault, and a non-faulty run which gives the same observation through the mask. Hence, for any k, A cannot be $((M, n), k)$-diagnosable. Thus A is not (M, n)-diagnosable which contradicts diagnosability of A.

Note that the above implies in particular that $n \geq 1$. We can now prove that G is n-colorizable. Let C be the color mapping defined by $C(v) = M(v)$. We need to prove that $C(u_i) \neq C(v_i)$ for any $(u_i, v_i) \in E$. This holds by construction of A_G and the fact that $M(u_i) \neq M(v_i)$ as shown above.

Only if part Assume G is n-colorizable. There exists a color mapping $C : V \to \{1, 2, \cdots, n\}$ s.t. if $(v, v') \in E$ then $C(v) \neq C(v')$. Define the mask M by $M(a) = C(a)$ for $a \in V$. We claim that A_G is $((M, n), 1)$-diagnosable (thus, also (M, n)-diagnosable). Assume on the contrary that A_G is not $((M, n), 1)$-diagnosable. Then there exist two runs $\rho \in \mathit{Faulty}_{\geq 1}(A_G)$ and $\rho' \in \mathit{NonFaulty}(A_G)$ such that $M^*(\boldsymbol{\pi}_{/\Sigma}(tr(\rho))) = M^*(\boldsymbol{\pi}_{/\Sigma}(tr(\rho')))$. As ρ is 1-faulty it must be of the form $\rho = v_1.\tau.u_1 \cdots u_i.f.v_i.\tau^k$ with $1 \leq i \leq j$ and $k \geq 0$. Notice that $M(a) \neq \varepsilon$ for all $a \in V$. Hence it implies $M^*(\boldsymbol{\pi}_{/\Sigma}(tr(\rho))) = M(v_1).M(u_1) \cdots M(u_i).M(v_i)$, and $|M^*(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))| = 2i$. Consequently, $|M^*(\boldsymbol{\pi}_{/\Sigma}(\rho'))| = 2i$.

The only possible such ρ' which is non-faulty is $\rho' = v_1.\tau.u_1 \cdots v_i.\tau.u_i$. Now, $M^*(\boldsymbol{\pi}_{/\Sigma}(tr(\rho))) = M^*(\boldsymbol{\pi}_{/\Sigma}(tr(\rho')))$, which implies $M(v_i) = M(u_i)$ i.e., $C(v_i) = C(u_i)$. But $(u_i, v_i) \in E$, and this contradicts the assumption that C is a valid coloring. □

5. Fault Diagnosis with Dynamic Observers

In this section we introduce *dynamic observers*. They can choose after each new observation the set of events they are going to watch for. To illustrate why dynamic observers can be useful consider the following example.

Example 5.1. (Dynamic Observation)
Assume we want to detect faults in automaton $\mathcal{B}$ of Fig. 6. A static diagnoser that observes $\Sigma = \{a, b\}$ works, however, no proper subset of Σ can be used to detect faults in $\mathcal{B}$. Thus the minimum cardinality of the set of observable events for diagnosing $\mathcal{B}$ is 2 i.e., a static observer will have to monitor two events during the execution of the DES $\mathcal{B}$. If we want to use a mask, the minimum-cardinality for a mask is

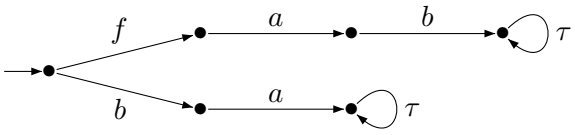


Figure 6. The automaton $\mathcal{B}$.

2 as well. This means that an observer will have to be receptive to at least two inputs at each point in time to detect a fault in $\mathcal{B}$. One can think of being receptive as switching on a device to sense an event. This consumes energy. We can be more efficient using a dynamic observer, that only turns on sensors when needed, thus saving energy. In the case of $\mathcal{B}$, this can be done as follows: in the beginning we only switch on the a-sensor; once an a occurs the a-sensor is switched off and the b-sensor is switched on. Compared to the previous diagnosers we use half as much energy.

5.1. Dynamic Observers

We formalize the above notion of dynamic observation using *observers*. The choice of the events to observe can depend on the choices the observer has made before and on the observations it has made. Moreover an observer may have *unbounded* memory.

Definition 5.1. (Observer)
An *observer* Obs over Σ is a deterministic labeled automaton[6] Obs $= (S, s_0, \Sigma, \delta, L)$, where S is a (possibly infinite) set of states, $s_0 \in S$ is the initial state, Σ is the set of observable events, $\delta : S \times \Sigma \to S$ is the transition function (a total function), and $L : S \to 2^\Sigma$ is a labeling function that specifies the set of events that the observer wishes to observe when it is at state s. We require for any state s and any

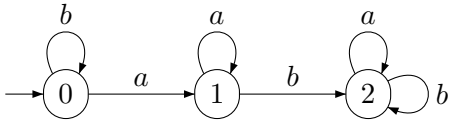
[6] We write $(S, s_0, \Sigma, \delta, L)$ instead of $((S, s_0, \Sigma, \delta, S, \emptyset), L)$.

$a \in \Sigma$, if $a \notin L(s)$ then $\delta(s,a) = s$: this means the observer does not change its state when an event it has chosen not to observe occurs. ■

As an observer is deterministic we use the notation $\delta(s_0, w)$ to denote the state s reached after reading the word w and $L(\delta(s_0, w))$ is the set of events Obs observes after w.

An observer implicitly defines a *transducer* that consumes an input event $a \in \Sigma$ and, depending on the current state s, either outputs a (when $a \in L(s)$) and moves to a new state $\delta(s,a)$, or outputs ε, (when $a \notin L(s)$) and remains in the same state waiting for a new event. Thus, an observer defines a mapping Obs from Σ^* to Σ^* (we use the same name "Obs" for the automaton and the mapping). Given a run ρ, $\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))$ is the output of the transducer on ρ. It is called the *observation* of ρ by Obs. We next provide an example of a particular case of observer which can be represented by a finite-state machine.

Example 5.2. Let Obs be the observer of Fig. 7. Obs maps the following inputs as follows: $\text{Obs}(baab) = ab$, $\text{Obs}(bababbaab) = ab$, $\text{Obs}(bbbbba) = a$ and $\text{Obs}(bbaaa) = a$. If Obs operates on the DES $\mathcal{B}$ of



$L(0) = \{a\}\ \ L(1) = \{b\}\ \ L(2) = \emptyset$

Figure 7. A finite-state observer Obs.

Fig. 6 and $\mathcal{B}$ generates $f.a.b$, Obs will have as input $\boldsymbol{\pi}_{/\Sigma}(f.a.b) = a.b$ with $\Sigma = \{a, b\}$. Consequently the observation of Obs is $\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(f.a.b)) = a.b$.

5.2. Checking Diagnosability with Dynamic Diagnosers

Definition 5.2. ((Obs, k)-diagnoser)
Let A be a finite automaton over $\Sigma^{\tau,f}$ and Obs be an observer over Σ. $D : \Sigma^* \to \{0,1\}$ is an (Obs, k)-*diagnoser* for A if

- $\forall \rho \in \mathit{NonFaulty}(A)$, $D(\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))) = 0$ and
- $\forall \rho \in \mathit{Faulty}_{\geq k}(A)$, $D(\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))) = 1$.

■

A is (Obs, k)-diagnosable if there is an (Obs, k)-diagnoser for A. A is Obs-diagnosable if there is some k such that A is (Obs, k)-diagnosable.

If a diagnoser always selects Σ as the set of observable events, it is a static observer and (Obs, k)-diagnosability amounts to the standard (Σ, k)-diagnosis problem [16]. In this case A is (Σ, k)-diagnosable iff equation (4) holds.

As for Σ-diagnosability, (Obs, k)-diagnosability can be stated as a language-based equality: A is (Obs, k)-diagnosable iff

$$\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(\mathit{Faulty}^{tr}_{\geq k}(A))) \cap \text{Obs}(\boldsymbol{\pi}_{/\Sigma}(\mathit{NonFaulty}^{tr}(A))) = \emptyset.$$

This follows directly from definition 5.2.

Problem 3. (Finite-State Obs-Diagnosability)

INPUT: Plant model $A = (Q, q_0, \Sigma^{\tau,f}, \rightarrow)$, Obs $= (S, s_0, \Sigma, \delta, L)$ a finite-state observer.
PROBLEM:

(A) Is A Obs-diagnosable?

(B) If the answer to (A) is "yes", compute the minimum k such that A is (Obs, k)-diagnosable.

Theorem 5.1. Problem 3 is in P.

Proof:
We first prove that Problem 3(A) is in P. The proof runs as follows: we build a *product* automaton[7] $A \otimes \text{Obs}$ such that: A is (Obs, k)-diagnosable $\iff$ $A \otimes \text{Obs}$ is (Σ, k)-diagnosable.

Let $A = (Q, q_0, \Sigma^{\tau,f}, \rightarrow)$ be a finite automaton and Obs $= (S, s_0, \Sigma, \delta, L)$ be a finite-state observer. Define the automaton $A \otimes \text{Obs} = (Q \times S, (q_0, s_0), \Sigma^{\tau,f}, \rightarrow)$ as follows:

- $(q, s) \xrightarrow{\beta} (q', s')$ iff $\exists \lambda \in \Sigma$ s.t. $q \xrightarrow{\lambda} q'$, $s' = \delta(s, \lambda)$ and $\beta = \lambda$ if $\lambda \in L(s)$, $\beta = \tau$ otherwise;
- $(q, s) \xrightarrow{\lambda} (q', s)$ iff $\exists \lambda \in \{\tau, f\}$ s.t. $q \xrightarrow{\lambda} q'$.

In this proof we use $\text{Obs}(\rho)$ instead of $\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))$ to simplify notations. To prove Theorem 5.1 we use the following lemmas:

Lemma 5.1.

1. Let $\rho \in \mathit{Faulty}_{\geq k}(A)$. There is a word $\nu \in \mathit{Faulty}^{tr}_{\geq k}(A \otimes \text{Obs})$ s.t. $\text{Obs}(\rho) = \boldsymbol{\pi}_{/\Sigma}(\nu)$.
2. Let $\rho' \in \mathit{NonFaulty}(A)$. There is a word $\nu' \in \mathit{NonFaulty}^{tr}(A \otimes \text{Obs})$ s.t. $\text{Obs}(\rho') = \boldsymbol{\pi}_{/\Sigma}(\nu')$.

Proof:
Let $r \in \mathit{Runs}(A)$ s.t. $r = q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$. By definition of $A \otimes \text{Obs}$, the run $\tilde{r} = (q_0, s_0) \xrightarrow{b_1} (q_1, s_1) \cdots (q_{n-1}, s_{n-1}) \xrightarrow{b_n} (q_n, s_n)$ such that for $1 \leq i \leq n$:

- if $a_i \in \{\tau, f\}$, $b_i = a_i$ and $s_i = s_{i-1}$;
- if $a_i \in L(s_{i-1})$, $b_i = a_i$ and $s_i = \delta(s_{i-1}, a_i)$;
- if $a_i \notin L(s_{i-1})$, $b_i = \tau$ and $s_i = s_{i-1}$;

is in $\mathit{Runs}(A \otimes \text{Obs})$. Moreover (i) $a_1 a_2 \cdots a_n \in \mathit{Faulty}^{tr}_{\geq k}(A) \iff b_1 b_2 \cdots b_n \in \mathit{Faulty}^{tr}_{\geq k}(A \otimes \text{Obs})$ and (ii) $\text{Obs}(r) = \boldsymbol{\pi}_{/\Sigma}(b_1 b_2 \cdots b_n)$ which means that $\text{Obs}(r) = \boldsymbol{\pi}_{/\Sigma}(tr(\tilde{r}))$.

By (i), if $r \in \mathit{Faulty}_{\geq k}(A)$, $tr(\tilde{r}) \in \mathit{Faulty}^{tr}_{\geq k}(A \otimes \text{Obs})$ and if $r \in \mathit{NonFaulty}(A)$, $tr(\tilde{r}) \in \mathit{NonFaulty}^{tr}(A \otimes \text{Obs})$ and in both cases $\text{Obs}(r) = \boldsymbol{\pi}_{/\Sigma}(tr(\tilde{r}))$ by (ii). □

Lemma 5.2.

[7] We use $\otimes$ to clearly distinguish this product from the synchronous product $\times$.

1. Let $\nu \in \mathit{Faulty}^{tr}_{\geq k}(A \otimes \text{Obs})$. There is a run $\rho \in \mathit{Faulty}_{\geq k}(A)$ s.t. $\text{Obs}(\rho) = \boldsymbol{\pi}_{/\Sigma}(\nu)$.
2. Let $\nu' \in \mathit{NonFaulty}^{tr}(A \otimes \text{Obs})$. There is a run $\rho' \in \mathit{NonFaulty}(A)$ s.t. $\text{Obs}(\rho') = \boldsymbol{\pi}_{/\Sigma}(\nu')$.

Proof:
This proof is the counterpart of the proof of Lemma 5.1. Let $r = (q_0, s_0) \xrightarrow{b_1} (q_1, s_1) \cdots (q_{n-1}, s_{n-1}) \xrightarrow{b_n} (q_n, s_n)$ be in $\mathit{Runs}(A \otimes \text{Obs})$. Define a run $\tilde{r} = q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$ with $1 \leq i \leq n$:

- if $b_i = f$, then $a_i = f$,
- if $b_i \in \Sigma$, then $a_i = b_i$,
- if $b_i = \tau$, choose some $a_i \in \{\tau\} \cup \Sigma \setminus L(s_{i-1})$ s.t. $q_{i-1} \xrightarrow{a_i} q_i$.

$\tilde{r}$ is a run in $\mathit{Runs}(A)$ and $\text{Obs}(\tilde{r}) = \boldsymbol{\pi}_{/\Sigma}(tr(r))$. It is then easy to see that Lemma 5.2 holds. □

Now assume A is (Obs, k)-diagnosable and $A \otimes \text{Obs}$ is not (Σ, k)-diagnosable. There are two words $\nu \in \mathit{Faulty}^{tr}_{\geq k}(A \otimes \text{Obs})$ and $\nu' \in \mathit{NonFaulty}^{tr}(A \otimes \text{Obs})$ s.t. $\boldsymbol{\pi}_{/\Sigma}(\nu) = \boldsymbol{\pi}_{/\Sigma}(\nu')$. By Lemma 5.2, there are two runs $\rho \in \mathit{Faulty}_{\geq k}(A)$ and $\rho' \in \mathit{NonFaulty}(A)$ s.t. $\text{Obs}(\rho) = \boldsymbol{\pi}_{/\Sigma}(\nu) = \boldsymbol{\pi}_{/\Sigma}(\nu') = \text{Obs}(\rho')$ and thus A is not (Obs, k)-diagnosable which is a contradiction.

Assume $A \otimes \text{Obs}$ is (Σ, k)-diagnosable and A is not (Obs, k)-diagnosable. There are two runs $\rho \in \mathit{Faulty}_{\geq k}(A)$ and $\rho' \in \mathit{NonFaulty}(A)$ with $\text{Obs}(\rho) = \text{Obs}(\rho')$. By Lemma 5.1, there are two words $\nu \in \mathit{Faulty}^{tr}_{\geq k}(A \otimes \text{Obs})$ and $\nu' \in \mathit{NonFaulty}^{tr}(A \otimes \text{Obs})$ s.t. $\text{Obs}(\rho) = \boldsymbol{\pi}_{/\Sigma}(\nu)$ and $\boldsymbol{\pi}_{/\Sigma}(\nu') = \text{Obs}(\rho')$ and thus $\boldsymbol{\pi}_{/\Sigma}(\nu) = \boldsymbol{\pi}_{/\Sigma}(\nu')$. This would imply that $A \otimes \text{Obs}$ is not (Σ, k)-diagnosable which is a contradiction.

The number of states of $A \otimes \text{Obs}$ is at most $|Q| \times |S|$ and the number of transitions is bounded by the number of transitions of A. Hence the size of the product is polynomial in the size of the input $|A| + |\text{Obs}|$. Checking that $A \otimes \text{Obs}$ is diagnosable can be done in polynomial time (section 3.1) thus Problem 3.(A) is in P.

To solve Problem 3(B), we can use the same algorithm as for finding the minimum k for a static diagnoser, explained in section 3.1. In this case we apply it to the product $A \otimes \text{Obs}$. The algorithm is polynomial in the size of $A \otimes \text{Obs}$, thus Problem 3(B) is also in P. This completes the proof. □

Example 5.3. Let $\mathcal{A}$ be the DES given in Fig. 2 and Obs the observer of Fig. 7. The product $\mathcal{A} \otimes \text{Obs}$ used in the above proof is given in Fig. 8. (Notice that this is different from the synchronized product (see section 2.4) $\mathcal{A} \times \text{Obs}$ since it uses the labeling information available in Obs.) Using an algorithm for checking Σ-diagnosability of $\mathcal{A} \otimes \text{Obs}$ we obtain that it is $(\Sigma, 2)$-diagnosable (and 2 is the minimum value). Hence $\mathcal{A}$ is $(\text{Obs}, 2)$-diagnosable with 2 the minimum value.

For Problem 3, we have assumed that an observer was given. It would be even better if we could *synthesize* an observer Obs such that the plant is Obs-diagnosable. Before attempting to synthesize such an observer, we should first check that the plant is Σ-diagnosable: if it is not, then obviously no such observer exists; if the plant is Σ-diagnosable, then the trivial observer that observes all events in Σ at all times works[8]. As a first step towards synthesizing non-trivial observers, we can attempt to compute the set of *all* valid observers, which includes the trivial one but also non-trivial ones (if they exist).

[8] Notice that this also shows that existence of an observer implies existence of a finite-state observer, since the trivial observer is finite-state.

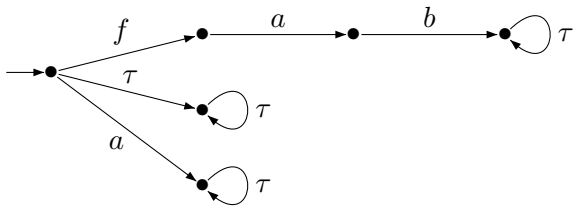


Figure 8. The product $\mathcal{A} \otimes \text{Obs}$ of the automaton of Fig. 2 and the observer of Fig. 7.

Problem 4. (Dynamic-Diagnosability)

INPUT: Plant model $A = (Q, q_0, \Sigma^{\tau,f}, \delta)$.
PROBLEM: Compute the set $\mathcal{O}$ of all observers such that A is Obs-diagnosable iff Obs $\in \mathcal{O}$.

We do not provide a solution to Problem 4 in this paper. The problem is left open and is part of our future work. Instead, we focus on the following problem:

Problem 5. (Dynamic-k-Diagnosability)

INPUT: Plant model $A = (Q, q_0, \Sigma^{\tau,f}, \delta)$, $k \in \mathbb{N}$.
PROBLEM: Compute a set of observers $\mathcal{O}$ such that for any observer Obs, A is (Obs, k)-diagnosable iff Obs $\in \mathcal{O}$. That is, $\mathcal{O}$ includes all observers (and only those) Obs for which A is (Obs, k)-diagnosable.

In the next subsection, we reduce Problem 5 to a *safety* control problem with partial information. Before giving a solution to Problem 5, we review some useful results on safety games that will be needed in the sequel.

5.3. Some Results on Safety Games Under Partial Observation

Computing controllers for system under partial observation has already been studied in [7, 13, 14]. These results apply to general alternating two-player games under partial observation and also for very powerful control objectives (Büchi control objectives). In the sequel we need to use results of the type of those presented in [7, 13, 14] but for a very particular type of two-player games, and only for safety objectives. This is why we provide hereafter the definitions and algorithms to solve this particular instance.

We model a control problem as a game problem between two players. In our setting, partial observation only comes from the fact that Player 1 cannot observe all Player 2 moves. Still Player 1 has to enforce a *safety* objective i.e., to force the system to stay in a designated set of (safe) states. Player 1 must also play according to the sequence of events it has observed so far: the strategy of Player 1 must be *trace-based*. In the sequel we formalize these assumptions and give some useful results on this type of games.

5.3.1. Games, Plays, Safety Objective

In each state of the game, it is either the turn of Player 1 to play or the turn of Player 2 to play, but not both (i.e., the moves of the two players alternate). The game starts in a Player 1 state. The rules of the game are as follows:

- Player 1 always plays one move (in the alphabet Σ_1) and hands it over to player 2;
- Player 2 can play two types of moves: invisible moves (τ is the invisible action) and visible moves (Σ_2-actions). If Player 2 plays an invisible move τ, it is again his turn to play. Otherwise if he plays a visible move, the turn switches back to Player 1.

Let $\uplus$ denote the union of disjoint sets.

Definition 5.3. (Two-Player τ-Games)
A *two-player τ-game* G is a tuple $(Q_1 \uplus Q_2, q_0, \Sigma_1 \uplus \Sigma_2^\tau, \delta)$ with:

- Q_i, $i = 1, 2$ is a finite set of states for player i;
- $q_0 \in Q_1$ is the *initial* state of the game;
- Σ_i is a finite set of visible actions for player $i, i = 1, 2$;
- $\delta \subseteq (Q_1 \times \Sigma_1 \times Q_2) \cup (Q_2 \times \{\tau\} \times Q_2) \cup (Q_2 \times \Sigma_2 \times Q_1)$ is the transition relation.

Remark 5.1. The type of δ is set according to the rules of the game.

We further assume that the game is deterministic w.r.t Player 1 moves, i.e., for all $q_1 \in Q_1, \sigma_1 \in \Sigma_1$ there is at most one state $q_2 \in Q_2$ such that $(q_1, \sigma_1, q_2) \in \delta$. For $(q, a, q') \in \delta$ we also use the notation $q\, a\, q'$ to write sequences of transitions in a compact manner. If G contains no τ transitions, G is an alternating full-observation two-player game (we use the term two-player game in this case).

Definition 5.4. (Play, Trace)
A *play* in G is a finite or infinite sequence

$$\rho = q_0 \ell_1 q_1 \cdots q_n \ell_{n+1} q_{n+1} \cdots$$

such that for each i, $q_i \xrightarrow{\ell_{i+1}} q_{i+1}$. We write $q_0 \xrightarrow{\ell_1 \ell_2 \cdots \ell_n} q_n$ if $q_0 \ell_1 q_1 \cdots \ell_n q_n$ is a finite play in G. We let $\mathit{Runs}(G)$ be the set of finite plays in G. $\mathit{Runs}_i(G), i = 1, 2$ are the sets of finite plays ending in a Q_i-state. The *state sequence* of a play ρ, denoted $\mathit{States}(\rho)$ is the sequence $q_0 q_1 \cdots q_n \cdots$.

We let $\rho(i)$ be the prefix of ρ up to state q_i, so $\rho(0) = q_0$, $\rho(n) = q_0 \ell_1 q_1 \cdots q_n$, and so on.

Definition 5.5. (Player 1 Strategy)
A strategy for Player 1 is a mapping $f : \mathit{Runs}_1(G) \to \Sigma_1$. A strategy f is *trace-based* if for all $\rho, \rho' \in \mathit{Runs}_1(G)$, $\mathit{tr}(\rho) = \mathit{tr}(\rho')$ implies $f(\rho) = f(\rho')$. A strategy f is *memoryless* if $\mathit{tgt}(\rho) = \mathit{tgt}(\rho')$ implies $f(\rho) = f(\rho')$.

Definition 5.6. (Outcome)
Given a strategy f for Player 1, the outcome, $Out(G, f)$, of the game G under (or controlled by) f is the set of states sequences[9] $\mathit{States}(\rho)$ for the plays $\rho = q_0 \ell_1 q_1 \cdots q_n \ell_{n+1} q_{n+1} \cdots$ such that for each $q_i \in Q_1$, $\ell_{i+1} = f(\rho(i))$.

[9] In the sequel we assume that the game objectives are states sequences and do not refer to the labels of the transitions.

Definition 5.7. (Safety Control Objective)
An *objective* ϕ for Player 1 is a subset of $(Q_1 \cup Q_2)^\omega \cup (Q_1 \cup Q_2)^*$. ϕ is a safety objective if there is a set $F \subseteq Q_1 \cup Q_2$ such that $\phi = F^\omega \cup F^*$ (F is the base set of *safe* states).

Given a pair (G, ϕ), a strategy f is *winning* if $Out(G, f) \subseteq \phi$. If G is a τ-game and ϕ a safety objective based on the set F, we say that the pair (G, F) is a *safety* game. A state q of G is winning if there is a winning strategy from q.

5.3.2. Safety Two-Player τ-Games

The usual control problem on two-player τ-games asks the following:

Problem 6. (Control Problem)

INPUT: A two-player τ-game G, a safety control objective F.
PROBLEM: Is there a winning strategy for (G, F) ?

To solve safety two-player τ-games we define the *Cpre* operator [17]:

$$\begin{aligned} \mathit{Cpre}(S) &= \{s \in Q_1 \,|\, \exists \sigma_1 \in \Sigma_1,\ s.t.\ \delta(s, \sigma_1) \in S\} && (5) \\ &\cup \{s \in Q_2 \,|\, \forall \sigma_2 \in \Sigma_2^\tau,\ \delta(s, \sigma_2) \subseteq S\} && (6) \end{aligned}$$

It is well-known [17] that iterating *Cpre* and computing the fix-point $\mathit{Cpre}^*(F)$ gives the set of winning states[10] of the game. In case G is finite this computation terminates and can be done in linear time for safety games [17]. If $q_0 \in \mathit{Cpre}^*(F)$, there is a strategy for Player 1 to win. Moreover, for this type of games, *memoryless* strategies are sufficient to win. Indeed, as we can *observe* the state of the game, τ transitions are not totally unobservable (or invisible) and thus knowing the current state gives some useful information. Also there is a *most permissive strategy*[11] $\mathcal{F} : Q_1 \cap \mathit{Cpre}^*(F) \to 2^{\Sigma_1} \setminus \emptyset$ defined by: $\mathcal{F}(q) = \{\sigma \,|\, \delta(q, \sigma) \in \mathit{Cpre}^*(F)\}$. This is the most permissive strategy in the following sense: f is a winning strategy for (G, F) iff for any finite run $\rho \in Out(G, f)$ ending in Q_1-state, $f(\rho) \in \mathcal{F}(\mathit{tgt}(\rho))$, i.e., every move defined by f is a move of the most permissive strategy and *vice versa*.

In our setting, τ is not observable. What we should ask for to win a game under partial observation, is to base our choice of move on the sole basis on the observable moves that have occurred so far. The problem we want to address is the following:

Problem 7. (Trace-Based Control Problem)

INPUT: a game G, a safety objective F.
PROBLEM: Is there a trace-based winning strategy for (G, F) ?

This problem is more demanding than the usual Control Problem (Problem 6) that asks only for a winning strategy for Player 1, i.e., a strategy in which full observation of the state is assumed. To solve Problem 7, we reduce it to a full-observation two-player game (using a construction like the one given in [7]).

[10] Notice that by definition of *Cpre*, Player 1 cannot win by refusing to play.
[11] According to Definition 5.5, it is not a strategy as it prescribes a set of moves for a given state instead of one move.

To do this, we first define the following operator for $\sigma \in \Sigma_2^\tau$, $s \in Q_2$,

$$\mathit{Next}_2(s,\sigma) = \{s' \mid s \xrightarrow{\tau^*} s_1 \xrightarrow{\sigma} s'\} \tag{7}$$

It follows that if $\sigma \in \Sigma_2$, $\mathit{Next}_2(s,\sigma) \subseteq Q_1$ and if $\sigma = \tau$, $\mathit{Next}_2(s,\sigma) \subseteq Q_2$. Let $\sigma \in \Sigma_1$ and $Q \subseteq Q_1$. We define

$$\mathit{Next}_1(Q,\sigma) = \{s' \in Q_2 \mid s' = \delta(s,\sigma) \textit{ with } s \in Q\} \tag{8}$$

To solve (G,F), we build a full-observation two player game (G_H,F_H) (which contains no τ transitions) such that:

Theorem 5.2. There is a winning strategy in (G_H,F_H) iff there is a trace-based winning strategy in (G,F).

The proof of Theorem 5.2 is given in Appendix A. The definition of the game (G_H,F_H) is as follows:

Definition 5.8. Assume $G = (Q_1 \uplus Q_2, q_0, \Sigma_1 \uplus \Sigma_2^\tau \delta)$. The game $G_H = (W, s_0, \Sigma', \Delta)$ is defined by:

- $W = W_1 \uplus W_2$ with $W_1 = 2^{Q_1} \cup \{\bot_1\}$ are the Player 1 states, $W_2 = 2^{Q_2} \cup \{\bot_2\}$ are Player 2 states;
- $s_0 = \{q_0\}$,
- $\Sigma' = \Sigma_1 \cup \Sigma_2^u$ where u is a fresh name, Σ_1 are Player 1 moves, and Σ_2^u Player 2 moves;
- $\Delta \subseteq (W_1 \times \Sigma_1 \times W_2) \cup (W_2 \times \Sigma_2^u \times W_1)$ is defined by: $(S,\sigma,S') \in \Delta$ iff one of the three conditions holds:
  - C_1: $S \subseteq Q_1, \sigma \in \Sigma_1$ and $S' = \mathit{Next}_1(S,\sigma)$ if for all $s \in S, \sigma \in \mathit{Enabled}(s)$ and otherwise $S' = \bot_2$;
  - C_2: $S \subseteq Q_2, \sigma \in \Sigma_2$, $S' = \mathit{Next}_2(S,\sigma)$ and $S' \neq \emptyset$;
  - C_3: $S \subseteq Q_2, \sigma = u$, $\mathit{Next}_2(S,\tau) \cap (W \setminus F) \neq \emptyset$ and $S' = \bot_1$.

We let $F_H = \{Q \in S \mid Q \subseteq F\}$ i.e., F_H is the set of safe states for G_H. $\bot_1$ and $\bot_2$ are not safe states. (G_H,F_H) is a safety game as well. Notice also that G_H is a full-observation turn-based two-player game in which the moves of the two players alternate. To solve Problem 7 we can first solve the game (G_H,F_H). This can be done in linear time in the size of (G_H,F_H). From this obtain obtain an algorithm for Problem 7 i.e., compute the set of winning states of G_H. As the size of G_H is exponential in the size G:

Theorem 5.3. Problem 7 is in EXPTIME.

Indeed, to solve Problem 7, we compute the set of winning states W_H of (G_H,F_H) and check that the initial state of G_H is winning. For safety games like G_H, with full observation, we can also compute the most permissive strategy as emphasised earlier. Given G_H and F_H let $\mathcal{F}_H$ be the most permissive strategy. This memoryless (state-based) strategy is given [17] by: let $q \in W_1$, $\sigma \in \mathcal{F}_H(q)$ if

$\mathit{Next}_1(q,\sigma) \in W_H$. In other words, the most permissive strategy gives, for each state, the set of moves that keep the system into the set of winning states W_H.

Given $\mathcal{F}_H$, we can build a strategy for G. Let $\beta(w)$ be the (unique) state s' s.t. $q_0 \xrightarrow{w} s'$ in G_H. We define the mapping $\mathcal{F}$ on $\mathit{Runs}_1(G)$ by: $\mathcal{F}(\rho) = \mathcal{F}_H(\mathit{tgt}(\beta(\mathit{tr}(\rho))))$. Intuitively, this means that, given a run ρ in G ending in a Q_1-state, to decide what to play, we compute the (unique) state $q = \mathit{tgt}(\beta(\mathit{tr}(\rho)))$ that can be reached in G_H; then we can play any move from the ones allowed by the most permissive strategy $\mathcal{F}_H(q)$.

Theorem 5.4. $\mathcal{F}$ is the most permissive trace-based strategy for G.

Proof:
First $\mathcal{F}$ is trace-based[12]. Let f be a trace-based winning strategy. We define f_H as in the *If* part proof of Theorem 5.2. f_H is winning and thus for any run w, we have $f_H(w) \in \mathcal{F}_H(\mathit{tgt}(w))$. By definition of f_H, $f(\rho) = f_H(\beta(\mathit{tr}(\rho)))$ and $f(\rho) \in \mathcal{F}_H(\mathit{tgt}(\beta(\mathit{tr}(\rho)))) = \mathcal{F}(\rho)$.

Now assume f is a trace-based strategy such that for each run $\rho \in \mathit{Runs}_1(G)$, $f(\rho) \in \mathcal{F}(\rho)$. Build f_H as before. f_H is winning. Indeed, $f_H(w) = f(\rho)$ for any ρ s.t. $\mathit{tr}(\rho) = \mathit{tr}(w)$. Hence, $f_H(w) \in \mathcal{F}_H(\mathit{tgt}(w))$ and thus f_H is winning. Now from f_H build a strategy $\tilde{f}$ as in the *Only If* part of the proof of Theorem 5.2. It turns out that $\tilde{f} = f$ and as $\tilde{f}$ is winning, f is winning. □

Corollary 5.1. The most permissive trace-based strategy $\mathcal{F}$ for (G,F) can be effectively computed and is represented by an automaton which has at most an exponential number of states in the size of G.

This follows from the fact that G_H is exponential in the size of G. The most permissive trace-based strategy is obtained from G_H by removing from each state q the transitions that are not in $\mathcal{F}_H(q)$.

5.4. Problem 5 as a Game Problem

To solve Problem 5 we reduce it to a *safety* 2-player τ-game. In short, the reduction we propose is the following:

- Player 1 chooses the set of events it wishes to observe, then it hands over to Player 2;
- Player 2 chooses an event and tries to produce a run which is the observation of both a k-faulty run and a non-faulty run.

Player 2 wins if he can produce such a run. Otherwise Player 1 wins. Player 2 has complete information of Player 1's moves (i.e., it can observe the sets that Player 1 chooses to observe). Player 1, on the other hand, only has partial information of Player 2's moves because not all events are observable (details follow). Let $A = (Q, q_0, \Sigma^{\tau,f}, \rightarrow)$ be a finite automaton. To define the game, we use two copies of automaton A: A_1^k and A_2. The accepting states of A_1^k are those corresponding to runs of A which are faulty and where more than k steps occurred after the fault. A_2 is a copy of A where the f-transitions have been removed. The game we are going to play is the following (see Fig. 9, Player 1 states are depicted with square boxes and Player 2 states with round shapes):

[12] Even if $\mathcal{F}$ is not strictly speaking a strategy, the trace-based property extends to sets of moves with set equality.

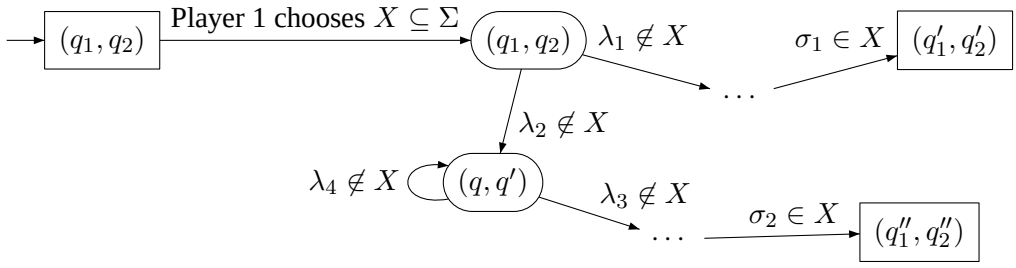


Figure 9. Game reduction for problem 5.

1. the game starts in a state (q_1, q_2) corresponding to the initial states the automata A_1^k and A_2. Initially, it is Player 1's turn to play. Player 1 chooses a set of events he is going to observe i.e., a subset X of Σ and hands it over to Player 2;

2. assume the automata A_1^k and A_2 are in states (q_1, q_2). Player 2 can change the state of A_1^k and A_2 by:

   (a) firing an action which is not in X (like $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ in Fig. 9): it is an unobservable action and thus, it can be taken independently by either A_1^k or A_2 (they do not need to synchronise). In this case a new state (q, q') is reached and Player 2 can play again from this state. Notice that Player 2's moves do not change the set X;

   (b) firing an action in X (like σ_1, σ_2 in Fig. 9): to do this both A_1^k and A_2 must be in a state where such an action (e.g., σ_1, σ_2) is enabled (synchronization); after the action is fired a new state (q'_1, q'_2) is reached: now it is Player 1's turn to play, and the game continues as in step 1 above from the new state (q'_1, q'_2).

Player 2 wins if he can reach a state (q_1, q_2) in the product of A_1^k and A_2 where q_1 is an accepting state of A_1^k (this means that Player 1 wins if it can avoid ad infinitum this set of states). In this sense this is a safety game for Player 1 (and a reachability game for Player 2). Formally, the game $G_A = (S_1 \uplus S_2, s_0, \Sigma_1 \uplus \Sigma_2, \delta_A)$ is defined as follows ($\uplus$ denotes union of disjoint sets):

- $S_1 = (Q \times \{-1, \cdots, k\}) \times Q$ is the set of Player 1 states; a state $((q_1, j), q_2) \in S_1$ indicates that A_1^k is in state q_1, j steps have occurred after a fault, and q_2 is the current state of A_2. If no fault has occurred, $j = -1$ and if more than k steps occurred after the fault, we use $j = k$.

- $S_2 = (Q \times \{-1, \cdots, k\}) \times Q \times 2^\Sigma$ is the set of Player 2 states. For a state $((q_1, j), q_2, X) \in S_2$, the triple $((q_1, j), q_2)$ has the same meaning as for S_1, and X is the set of moves Player 1 has chosen to observe on its last move.

- $s_0 = ((q_0, -1), q_0)$ is the initial state of the game belonging to Player 1;

- $\Sigma_1 = 2^\Sigma$ is the set of moves of Player 1; $\Sigma_2 = \Sigma^\tau$ is the set of moves of Player 2 (as we encode the fault into the state, we do not need to distinguish f from τ).

- the transition relation $\delta_A \subseteq (S_1 \times \Sigma_1 \times S_2) \cup (S_2 \times \{\tau\} \times S_2) \cup (S_2 \times \Sigma \times S_1)$ is defined by:

– Player 1 moves: let $\sigma \in \Sigma_1$ and $s_1 \in S_1$. Then $(s_1, \sigma, (s_1, \sigma)) \in \delta_A$.
– Player 2 moves: a move of Player 2 is either a silent move (τ) i.e., a move of A_1^k or A_2 or a joint move of A_1^k and A_2 with an observable action in X. Consequently, a *silent* move $((q_1, i), q_2, X), \tau, (q_1', j), q_2', X))$ is in δ_A if one of the following conditions holds:

  1. either $q_2' = q_2$, $q_1 \xrightarrow{\ell} q_1'$ is a step of A_1^k, $\ell \notin X$, and if $i \geq 0$ then $j = \min(i+1, k)$; if $i = -1$ and $\ell = f$, then $j = 0$ otherwise $j = i$.
  2. either $q_1' = q_1$, $q_2 \xrightarrow{\ell} q_2'$ is a step of A_2, $\ell \notin X$ (and $\ell \neq f$), and if $i \geq 0$ then $j = \min(i+1, k)$, otherwise $j = i$.

  A *visible* move can be taken by Player 2 if both A_1^k and A_2 agree on doing such a move. In this case the game proceeds to a Player 1 state: $((q_1, i), q_2, X), \ell, ((q_1', j), q_2')) \in \delta_A$ if $\ell \in X$, $q_1 \xrightarrow{\ell} q_1'$ is a step of A_1^k, $q_2 \xrightarrow{\ell} q_2'$ is a step of A_2, and if $i \geq 0$ then $j = \min(i+1, k)$, otherwise $j = i$.

We can show that for any observer O s.t. A is (O, k)-diagnosable, there is a strategy $f(O)$ for Player 1 in G_A s.t. $f(O)$ is *trace-based* and winning. A *strategy* for Player 1 is a mapping $f : \mathit{Runs}(G_A) \to \Sigma_1$ that associates a move $f(\rho)$ in Σ_1 with each run ρ in G_A that ends in an S_1-state. Remind that a strategy f is trace-based (see section 5.3 for details), if given two runs ρ, ρ', if $\mathit{tr}(\rho) = \mathit{tr}(\rho')$ then $f(\rho) = f(\rho')$. Conversely, for any trace-based winning strategy f (for Player 1), we can build an observer $O(f)$ s.t. A is $(O(f), k)$-diagnosable.

Let $O = (S, s_0, \Sigma, \delta, L)$ be an observer for A. We define the strategy $f(O)$ on finite runs of G_A ending in a Player 1 state by: $f(O)(\rho) = L(\delta(s_0, \boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho))))$. The intuition is that we take the run ρ in G_A, take the trace of ρ (choices of Player 1 and moves of Player 2) and remove the choices of Player 1. This gives a word in Σ^*. The strategy for Player 1 for ρ is the set of events the observer O chooses to observe after reading $\boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho))$ i.e., $L(\delta(s_0, \boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho))))$.

Theorem 5.5. Let O be an observer s.t. A is (O, k)-diagnosable. Then $f(O)$ is a trace-based winning strategy in G_A.

Proof:
First $f(O)$ is trace-based by definition. We have to prove that $f(O)$ is winning. We denote $\mathit{Out}(G, f)$ the set of outcomes i.e., the set of possible runs of a game G when the strategy f is played by Player 1 (see section 5.3 for a formal definition of $\mathit{Out}(G_A, f)$). Assume on the contrary that $f(O)$ is not winning. This implies that there is a run ρ in $\mathit{Out}(G_A, f(O))$ as defined by equations (9–11) below: Each step i of the run given by one of equations (9–11) consists of a choice of Player 1 (X_i move) followed by a number of moves by Player 2 (λ_i^j actions). The last state encountered in ρ, $((q_n^1(\alpha), k_n(\alpha)), q_n^2(\alpha), X_n)$ is a losing state for Player 1, which means that $k_n(\alpha) = k$, by definition of losing states in G_A. From the run ρ, we can build two runs ν and ν' defined by equations (12) and (13): By definition of G_A, each λ_i^j is either a common visible action of A_1^k and A_2 and it is in Σ, or a silent action (τ) i.e., it actually corresponds to an action of A_1^k or A_2 that is not in the current set of visible actions X_i. To retrieve a run in A_1^k (resp. A_2) from ν (resp. ν') we can safely remove the unobservable actions of A_2 (resp. A_1^k) because they leave the state of A_1^k (resp. A_2) unchanged. This way we obtain two runs $\tilde{\nu}$ and $\tilde{\nu}'$, which, when interleaved give a run of $A_1^k \times A_2$. By definition of G_A, $\tilde{\nu} \in \mathit{Faulty}_{\geq k}(A)$ and $\tilde{\nu}' \in \mathit{NonFaulty}(A)$. We claim that $O(\tilde{\nu}) = O(\tilde{\nu}')$. Indeed, each part of the runs from $q_i^1 \cdots q_{i+1}^1$ and $q_i^2 \cdots q_{i+1}^2$ yields the

$$\begin{aligned}\rho \quad = \quad & (q_0^1,-1), q_0^2 \xrightarrow{X_0} (q_0^1,0), q_0^2, X_0 \xrightarrow{\lambda_0^1} (q_0^1(1),k_0(1)), q_0^2(1), X_0 \cdots \\ & (q_0^1(j),k_0(j)), q_0^2(j), X_0 \cdots \xrightarrow{\lambda_0^{n_0}} \end{aligned} \tag{9}$$

$$\begin{aligned}& (q_1^1,k_1), q_1^2 \xrightarrow{X_1} (q_1^1,k_1), q_1^2, X_1 \xrightarrow{\lambda_1^1} (q_1^1(1),k_1(1)), q_1^2(1), X_1 \cdots \\ & (q_1^1(j),k_1(j)), q_1^2(j), X_1 \cdots \xrightarrow{\lambda_1^{n_1}} \end{aligned} \tag{10}$$

$$\vdots \tag{...}$$

$$\begin{aligned}& (q_n^1,k_n), q_n^2 \xrightarrow{X_n} (q_n^1,k_n), q_n^2, X_n \cdots (q_n^1(j),k_n(j)), q_n^2(j), X_n \cdots \\ & \xrightarrow{\lambda_0^\alpha} (q_n^1(\alpha),k_n(\alpha)), q_n^2(\alpha), X_n \end{aligned} \tag{11}$$

$$\nu \; = \; q_0^1 \xrightarrow{\lambda_0^1} q_0^1(1) \xrightarrow{\lambda_0^2} \cdots \xrightarrow{\lambda_0^{n0}} q_1^1 \xrightarrow{\lambda_1^1} \cdots \xrightarrow{\lambda_1^{n_1}} q_2^1 \cdots q_n^1 \xrightarrow{\lambda_n^1} \cdots \xrightarrow{\lambda_n^\alpha} q_n^1(\alpha) \tag{12}$$

$$\nu' \; = \; q_0^2 \xrightarrow{\lambda_0^1} q_0^2(1) \xrightarrow{\lambda_0^2} \cdots \xrightarrow{\lambda_0^{n0}} q_1^2 \xrightarrow{\lambda_1^1} \cdots \xrightarrow{\lambda_1^{n_1}} q_2^2 \cdots q_n^2 \xrightarrow{\lambda_n^1} \cdots \xrightarrow{\lambda_n^\alpha} q_n^2(\alpha) \tag{13}$$

same observation by O: it is the sequence of events $\lambda_{j_1} \cdots \lambda_{j_{n_i}}$ s.t. each λ_{j_l} is a letter of both A_1^k and A_2 and is in X_i. As there are two runs $\tilde{\nu} \in \mathit{Faulty}_{\geq k}(A)$ and $\tilde{\nu}' \in \mathit{NonFaulty}(A)$ with the same observation, A is not (O,k)-diagnosable which contradicts the assumption. Hence $f(O)$ must be winning. □

Conversely, with each trace-based strategy f of the game G_A we can associate an automaton $O(f) = (S, s_0, \Sigma, \delta, L)$ defined by:

- $S = \{\boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho)) \mid \rho \in \mathit{Out}(G_A, f) \text{ and } \mathit{tgt}(\rho) \in S_1\}$;
- $s_0 = \varepsilon$;
- $\delta(v,\ell) = v'$ if $v \in S$, $v' = v.\ell$ and there is a run $\rho \in \mathit{Out}(G_A, f)$ with $\rho = q_0 \xrightarrow{X_0} q_0^1 \xrightarrow{\tau^*} q_0^{n_0} \xrightarrow{\lambda_1} q_1 \xrightarrow{X_1} q_1^1 \xrightarrow{\tau^*} q_1^{n_1} \xrightarrow{\lambda_2} q_2 \cdots q_{k_1} \xrightarrow{\tau^*} q_{k-1}^{n_{k-1}} \xrightarrow{\lambda_k} q_k$ with each $q_i \in S_1$, $q_i^j \in S_2$, $v = \boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho))$, and $\rho \xrightarrow{X_k} q_k^1 \xrightarrow{\tau^*} q_k^{n_k} \xrightarrow{\ell} q_{k+1}$ with $q_{k+1} \in S_1$, $\ell \in X_k$.
  $\delta(v,l) = v$ if $v \in S$ and $\ell \notin f(\rho)$;
- $L(v) = f(\rho)$ if $v = \boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho))$.

Lemma 5.3. $O(f)$ is an observer.

Proof:
We first have to prove that $O(f)$ (more precisely L) is well defined. Assume $v = \boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho))$ and $v' = \boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho'))$. As f is trace-based, $f(\rho) = f(\rho')$ and there is a unique value for $L(v)$.

We also have to prove that the last requirement of Definition 5.1 is satisfied i.e., if $a \notin L(s)$ then $\delta(s,a) = s$. If $\ell \notin L(v)$, then $\ell \notin f(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))$ for any ρ s.t. $v = \boldsymbol{\pi}_{/\Sigma}(tr(\rho))$ because f is trace-based. Thus $\delta(v,\ell) = v$. □

Theorem 5.6. Let f be a trace-based winning strategy in G_A. Then A is $(O(f),k)$-diagnosable.

Proof:
Assume A is not $(O(f),k)$-diagnosable. There are two runs $\nu \in \mathit{Faulty}_{\geq k}(A)$ and $\nu' \in \mathit{NonFaulty}(A)$ s.t. $O(f)(\boldsymbol{\pi}_{/\Sigma}(tr(\nu))) = O(f)(\boldsymbol{\pi}_{/\Sigma}(tr(\nu')))$. Let $\tilde{\nu}$ (resp. $\tilde{\nu'}$) be the sequence of labels that appear in ν (resp. ν'). We can write $\tilde{\nu}$ and $\tilde{\nu'}$ in the form

$$\tilde{\nu} = w_{-1}\lambda_0 w_0 \lambda_1 w_1 \cdots \lambda_n w_n \quad (14)$$

$$\tilde{\nu'} = w'_{-1}\lambda_0 w'_0 \lambda_1 w'_1 \cdots \lambda_n w'_n \quad (15)$$

with $w_i, w'_i \in (\Sigma \setminus O(f)(\lambda_0\lambda_1\cdots\lambda_i)^*$ for $i \geq 0$ and $w_{-1}, w'_{-1} \in (\Sigma \setminus O(f)(\varepsilon))^*$, and $\lambda_{i+1} \in O(f)(\lambda_0\lambda_1\cdots\lambda_i)$. We build a run in $Out(G_A, f)$ as follows:

1. Player 1 chooses the set $X_0 = O(f)(\varepsilon)$ which is by definition equal to $f((q_0^1,-1),q_0^2)$ where $((q_0^1,-1),q_0^2)$ is the initial state of the game.
2. Player 2 plays the sequence of actions in w_1 and w'_1 synchronizing on the common actions of w_1 and w'_1. The game moves through S_2 states because each action is an invisible move. Finally Player 2 chooses $\lambda_0 \in O(f)(\varepsilon)$. The game reaches a new S_1-state $((q_1^1,k_1),q_1^2)$.
3. from $((q_1^1,k_1),q_1^2)$, the strategy f is to play X_1 which by definition is $O(f)(\lambda_0)$. Thus Player 2 can play the sequence of actions given in w_2 and w'_2 synchronizing again on common action. In the end, Player 2 plays $\lambda_1 \in O(f)(\lambda_0)$.

We can iterate the previous algorithm and build a run in $Out(G_A, f)$ that reaches a state $((q_n^1,k_n),q_n^2)$ with $k_n = k$ and thus $Out(G_A, f)$ contains a losing run. Hence f is not winning which contradicts the assumption. This way we conclude that A is $(O(f),k)$-diagnosable. □

The result on G_A (section 5.3) is that, if there is a winning trace-based strategy for Player 1, then there is a most permissive strategy $\mathcal{F}_A$ which has finite memory. It can be represented by a finite automaton $S_{\mathcal{F}_A} = (W_1 \uplus W_2, s_0, \Sigma \cup 2^\Sigma, \Delta_A)$ s.t. $\Delta_A \subseteq (W_1 \times 2^\Sigma \times W_2) \cup (W_2 \times \Sigma \times W_1)$ which has size exponential in the size of G_A. For a given run $\rho \in (\Sigma \cup 2^\Sigma)^*$ ending in a W_1-state, we have $\mathcal{F}_A(w) = \mathit{Enabled}(\Delta_A(s_0,w))$.

5.5. Most Permissive Observer

We now define the notion of *most permissive* observer and show the existence of a most permissive observer for a system in case A is diagnosable. $\mathcal{F}_A$ is the mapping defined at the end of the previous section.

For an observer $O = (S, s_0, \Sigma, \delta, L)$ and $w \in \Sigma^*$ we let $L(w)$ be the set $L(\delta(s_0,w))$: this is the set of events O chooses to observe on input w. Given a word $\rho \in \boldsymbol{\pi}_{/\Sigma}(\mathcal{L}(A))$, we recall that $O(\rho)$ is the observation of ρ by O. Assume $O(\rho) = a_0 \cdots a_k$. Let $\overline{\rho} = L(\varepsilon).a_0.L(a_0).a_1.\cdots a_k.L(O(\rho)(k))$ i.e., $\overline{\rho}$

contains the history of what O has chosen to observe at each step and the events that occurred after each choice.

Let $\mathcal{O} : 2^{\Sigma}.(\Sigma^{\tau} \times 2^{\Sigma})^{+} \to 2^{2^{\Sigma}}$. By definition $\mathcal{O}$ is the most permissive observer for (A, k) if the following holds:

$$O = (S, s_0, \Sigma, \delta, L) \text{ is an observer and } A \text{ is } (O, k)\text{-diagnosable} \iff \forall w \in \Sigma^*, L(\delta(s_0, w)) \in \mathcal{O}(\overline{w})$$

The definition of the most permissive observer states that:

- any good observer O (one such that A is (O, k)-diagnosable) must choose a set of observable events in $\mathcal{O}(\overline{w})$ on input w;
- if an observer chooses its set of observable events in $\mathcal{O}(\overline{w})$ on input w, then it is a good observer.

Assume A is (Σ, k)-diagnosable. Then there is an observer O s.t. A is (O, k)-diagnosable because the constant observer that observes Σ is a solution. By Theorem 5.5, there is a trace-based winning strategy for Player 1 in G_A. As said at the end of the previous subsection, in this case there is a most permissive trace-based winning strategy which is $\mathcal{F}_A$.

Theorem 5.7. $\mathcal{F}_A$ is the most permissive observer.

Proof:
Let $O = (S, s_0, \Sigma, \delta, L)$ be an observer such that A is (O, k)-diagnosable. We have to prove that $L(\delta(s_0, w)) \in \mathcal{F}_A(\overline{w})$ for any $w \in \Sigma^*$. By Theorem 5.5, the strategy $f(O)$ is a winning trace-based strategy and this implies that $f(O)(\nu) \in \mathcal{F}_A(\nu)$ for any run ν of G_A. By definition of $\overline{w}$, $\boldsymbol{\pi}_{/\Sigma}(\overline{w}) = w$. By definition of $f(O)$, $f(O)(\overline{w}) = L(\delta(s_0, \boldsymbol{\pi}_{/\Sigma}(tr(\overline{w})))) = L(\delta(s_0, w))$ and thus $L(\delta(s_0, w)) \in \mathcal{F}_A(\overline{w})$.

Conversely, assume O is such that $\forall w \in \Sigma^*, L(s_0, w) \in \mathcal{F}_A(\overline{w})$. We have to prove that A is (O, k)-diagnosable. Again, we build $f(O)$. As before, $f(O)$ is a winning trace-based strategy in G_A and thus $O(f(O))$ is such that A is $(O(f(O)), k)$-diagnosable by Theorem 5.6. Assume $O(f(O)) = (S', s'_0, \Sigma, \delta', L'))$. By construction of $O(f(O))$, $L'(\delta'(s'_0, w)) = f(O)(\rho)$ if $w = \boldsymbol{\pi}_{/\Sigma}(tr(\rho))$. Hence $O(f(O)) = O$ and A is (O, k)-diagnosable. □

This enables us to solve Problem 5 and compute a finite representation of the set $\mathcal{O}$ of all observers such that A is (O, k)-diagnosable iff $O \in \mathcal{O}$.

Computing $\mathcal{F}_A$ can be done in $O(2^{|G_A|})$ (section 5.3). The size of G_A is quadratic in $|A|$, linear in the size of k, and exponential in the size of Σ i.e., $|G_A| = O(|A|^2 \times 2^{|\Sigma|} \times |k|)$. This means that computing $\mathcal{F}_A$ can be done in exponential time in the size of A and k and doubly exponential time in the size of Σ.

Example 5.4. For the automaton $\mathcal{A}$ of Fig. 2, we obtain the most permissive observer $\mathcal{F}_{\mathcal{A}}$ shown in Fig. 10. In the even states, depicted as a square, the observer chooses (non-deterministically) what to observe and in the odd states, depicted as a circle, it moves according to what it observes. For odd states we have not shown the component X that has last been picked up by the observer. X is the label of the unique incoming transition. Initially, the observer may choose to observe either $\{a, b\}$ or $\{a\}$. If it

observes a, then it moves to state 2, where it can choose any subset containing b. If it observes b instead, then it moves to state 4, where it can choose to observe anything, including the empty set.

We point out that from an odd state $(2k+1, X)$, outgoing transitions are labeled by elements of X. This does not mean that the DES under observation cannot do other actions than those in X: it might be able to do so but these actions are unobservable for the observer.

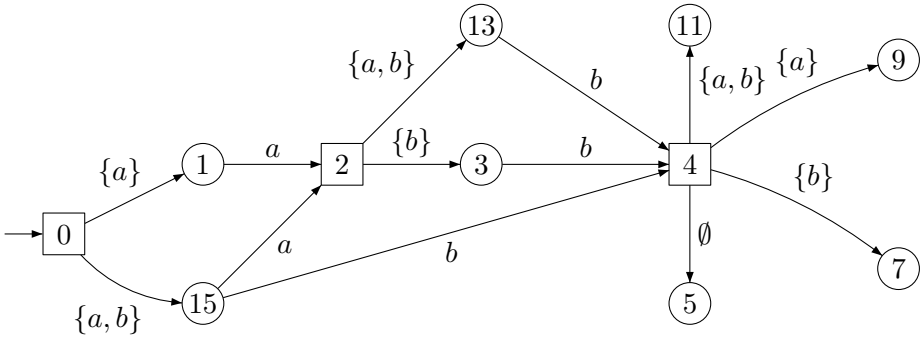


Figure 10. Most permissive observer for the automaton $\mathcal{A}$ of Fig. 2.

5.6. Computation of the Most Permissive Diagnoser

We conclude this section by showing how to compute a *generic* diagnoser. This diagnoser is the *most permissive dynamic* diagnoser and contains all the choices a dynamic diagnoser can make to be able to diagnose a plant.

The construction of the most permissive dynamic diagnoser follows easily from the construction of the most permissive observer and the construction of a synchronized product (using the $\otimes$ defined operator defined in section 5.2). The steps to build it are:

1. from the plant A, build $\tilde{A}$ which behaves like A but where the states which are after a faulty event are tagged i.e., each state of $\tilde{A}$ is of the form (q,k) where $k=0$ (non faulty states) or $k=1$ (faulty states);

2. determinize $\tilde{A}$ assuming f and τ are unobservable and obtain $\det(\tilde{A})$; the final states S of $\det(\tilde{A})$ are the the set of states of the form $S=\{(q_1,1),(q_2,1),\cdots,(q_n,1)\}$ i.e., all the states in S are faulty;

3. write the most permissive observer as a *non-deterministic* observer (like in Definition 5.1) and obtain O;

4. synchronize O and $\det(\tilde{A})$ to obtain the most permissive dynamic diagnoser: announce a fault when $O \otimes \det(\tilde{A})$ reaches a final state, and otherwise no fault can be announced.

We now define precisely the previous steps. Let $A=(Q,q_0,\Sigma^{\tau,f},\rightarrow)$ be a finite automaton over $\Sigma^{\tau,f}$.

Definition of det$(\tilde{A})$**.** Define $\tilde{A} = (Q \times \{0,1\}, (q_0, 0), \Sigma^\tau, \rightarrow)$ s.t.

- $(q,n) \xrightarrow{l}_\sim (q',n')$ iff $q \xrightarrow{l} q'$ and $l \in \Sigma$ and $n = n'$;
- $(q,n) \xrightarrow{\tau}_\sim (q',1)$ iff $q \xrightarrow{f} q'$, (n is set to 1 after a fault);
- $(q,n) \xrightarrow{\tau}_\sim (q',n)$ iff $q \xrightarrow{\tau} q'$.

We equip $\tilde{A}$ with the set of final states $F = Q \times \{1\}$. For the example of Fig. 2 page 7, $F = \{1,2,3\} \times \{1\}$. Let det$(\tilde{A})$ be the determinization of $\tilde{A}$. A state $S = \{s_1, s_2, \cdots, s_n\}$ in det$(\tilde{A})$ is final if every state s_i is final in $\tilde{A}$.

Definition of $O \otimes$ ***det***$(\tilde{A})$**.** To compute the most permissive dynamic diagnoser, we first write the most permissive observer as a non-deterministic observer i.e., following Definition 5.1.

To obtain a "generic" dynamic diagnoser it suffices to synchronize it (using the $\otimes$ operator defined in section 5.2). The most permissive (non-deterministic) dynamic diagnoser $D_{\mathcal{A}}$ for the automaton $\mathcal{A}$ of Fig. 2 is given in Fig. 11. The final state of $D_{\mathcal{A}}$ is F_A pictured with double-lines. $D_{\mathcal{A}}$ announces of fault

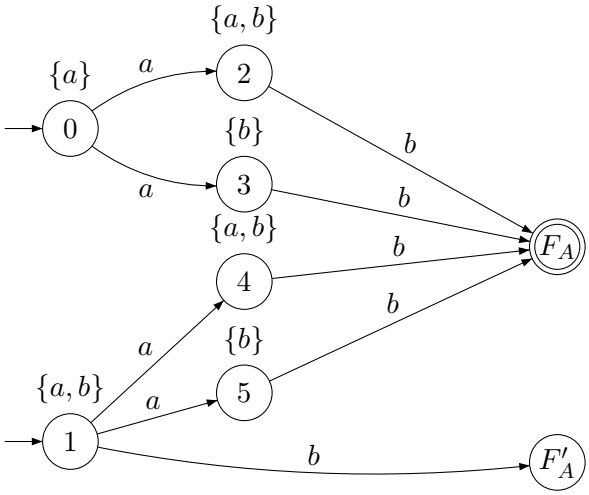


Figure 11. Most permissive dynamic diagnoser $D_{\mathcal{A}}$.

when it reaches its final state F_A. In state F_A and F'_A, $D_{\mathcal{A}}$ can choose to observe any subset of $\{a,b\}$ but no event can occur anymore. F_A is a final state whereas F'_A is not. Notice also that $D_{\mathcal{A}}$ starts either from 0 or 1. As the size of det$(\tilde{A})$ is also exponential in the size of A, $D_{\mathcal{A}}$ has size exponential in the size of A (and k) and doubly exponential in the size of Σ.

6. Optimal Dynamic Observers

In this section we define a notion of cost for observers. This will allow us to compare observers w.r.t. to this criterion and later on to synthesize an optimal observer.

6.1. Weighted Automata & Karp's Algorithm

Before introducing the optimal dynamic observer and diagnoser synthesis problem, we present a set of tools that we are going to use in that process. These deal with the notion of cost in a model of dynamic behaviors such as a finite automaton model. The notion of cost for automata has already been defined and algorithms to compute some optimal values related to this model are described in many papers. For our purposes, the model of *weighted automata* is appropriate. We recall here this model and the results of [12] which will be used later.

Definition 6.1. (Weighted Automaton)
A *weighted automaton* is a pair (A, w) s.t. $A = (Q, q_0, \Sigma, \delta)$ is a finite automaton and $w : Q \to \mathbb{N}$ associates a weight with each state. ■

Definition 6.2. (Mean Cost)
Let $\rho = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ be a run of A. The *mean cost* of ρ is

$$\mu(\rho) = \frac{1}{n+1} \times \sum_{i=0}^{n} w(q_i)\,.$$

■

We remind that the length of $\rho = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ is $|\rho| = n$. We assume that A is complete w.r.t. Σ (and $\Sigma \neq \emptyset$) and thus contains at least one run for any arbitrary length n. Let $\mathit{Runs}^n(A)$ be the set of runs of length n in $\mathit{Runs}(A)$. The *maximum mean-weight* of the runs of length n for A is $\nu(A, n) = \max\{\mu(\rho) \text{ for } \rho \in \mathit{Runs}^n(A)\}$. The *maximum mean weight* of A is $\nu(A) = \limsup_{n\to\infty} \nu(A, n)$. Actually the value $\nu(A)$ can be computed using Karp's maximum mean-weight cycle algorithm [12] on weighted graphs. If $c = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n$ is a cycle of A i.e., $s_0 = s_n$, the *mean weight* of the cycle c is $\mu(c) = \frac{1}{n+1} \cdot \sum_{i=0}^{n} w(s_i)$. The *maximum mean-weight cycle* of A is the value $\nu^*(A) = \max\{\mu(c) \text{ for } c \text{ a cycle of } A\}$. As stated in [24], for weighted automata, the mean-weight cycle value is the value that determines the mean-weight value (the transient behaviors of the system are not contributing to this value). It follows that $\nu(A) = \limsup_{n\to\infty} \nu(A, n) = \lim_{n\to\infty} \nu(A, n) = \nu^*(A)$.

The original Karp's maximum mean cycle algorithm [12] works for weighted automaton where the weights are on the edges. We give the version where weights are on vertices. Let $\nu^* = \max_c \mu(c)$ where c ranges over all cycles in A. A cycle c with $\mu(c) = \nu^*$ is a *maximum mean-weight cycle*. Let $D(q)$ be the weight of a most expensive path from q_0 to q and $D_k(q)$ be the weight of a most expensive path which has exactly k edges (if there is no such path $D_k(v) = -\infty$). Assume $|Q| = n$. Karp's algorithm is based on the fact that

$$\nu^* = \max_{q\in Q} \min_{0\le k\le n-1} \frac{D_n(q) - D_k(q)}{n-k}$$

The values $D_k(q)$ can be computed iteratively:

$$D_0(q_0) = w(q_0) \tag{16}$$
$$D_0(q) = -\infty \quad \text{for} \quad q \neq q_0 \tag{17}$$
$$D_{k+1}(q) = \max_{q\in\delta(q',a)} \{D_k(q') + w(q)\} \tag{18}$$

Thus for each state q we can compute $\min(q) = \min_{0 \leq k \leq n-1} \frac{D_n(q) - D_k(q)}{n-k}$ and then compute the value $\max_{q \in Q} \min(q)$ to obtain ν^*. This algorithm runs in $O(n.m)$ where $|Q| = n$ and $|\delta| = m$ (where $|\delta|$ denotes the number of transitions in δ). Improvements [5] can be made to this algorithm still the worst case run-time is $O(n.m)$.

6.2. Cost of a Dynamic Observer

Let Obs $= (S, s_0, \Sigma, \delta, L)$ be an observer and $A = (Q, q_0, \Sigma^{\tau,f}, \rightarrow)$. We would like to define a notion of *cost* for observers in order to select an optimal one among all of those which are valid, i.e., s.t. A is (Obs, k)-diagnosable. Intuitively this notion of cost should capture the fact that the more events we observe at each time, the more expensive it is.

Definition of Cost. Given a word $w = a_0 a_1 \cdots a_n$, we let $w(i) = a_0 \cdots a_i$ be the prefix up to a_i of w. In the sequel $\text{Obs}(w)(i)$ thus denotes the prefix up to the ith letter of $\text{Obs}(w)$.

There is not one way of defining a notion of cost for observers and we first discuss two different notions:

- the first one is to define the cost of a word w generated by the DES w.r.t. to $\text{Obs}(w)$:

$$\mathit{Cost}_1(w) = \frac{\sum_{i=0}^{i=n} |L(\delta(s_0, \text{Obs}(w)(i)))|}{n+1}$$

  with $n = |\text{Obs}(w)|$. Using the observer of Fig. 12, we obtain that $\mathit{Cost}_1(b^n.a) = \frac{1+0}{2} = \frac{1}{2}$. And this regardless of the value of n.

- the second one is to define the cost of w w.r.t. to w itself:

$$\mathit{Cost}_2(w) = \frac{\sum_{i=0}^{i=n} |L(\delta(s_0, w(i)))|}{n+1}$$

  with $n = |w|$. Using the observer of Fig. 12, we obtain $\mathit{Cost}_2(b^n.a) = \frac{n+1+0}{n+2} = \frac{n+1}{n+2}$. And by simple arithmetic, it is true that $\mathit{Cost}_2(b^n.a) < \mathit{Cost}_2(b^{n+1}.a)$.

The example of Fig. 12 shows that the two notions are different. In the sequel we will use the second one, Cost_2, because Cost_2 also captures the notion of the *time* we have been observing a set of events. Indeed, if the word b^{n+1} occurs, we have been observing the set $L(0)$ $n+1$ times in a logical time. It is natural that this is more expensive than observing $L(0)$ n times. Thus Cost_2 is more satisfying than abstracting away the length of the input word as in Cost_1.

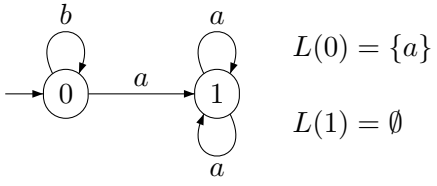


Figure 12. The Finite-State Observer Obs.

Cost of an Observer. We now show how to define and compute the cost of an observer Obs operating on a DES A.

Given a run $\rho \in \mathit{Runs}(A)$, the observer only processes $\boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho))$ (τ and f-transitions are not processed). To have a consistent notion of costs that takes into account the logical time elapsed from the beginning, we need to take into account one way or another the number of *steps* of ρ (the length of ρ) even if some of them are non observable. A simple way to do this is to consider that τ and f are now observable events, let's say u, but that the observer never chooses to observe them. Indeed we assume we have already checked that A is (Obs, k)-diagnosable, and the problem is now to compute the cost of the observer we have used.

Definition 6.3. (Cost of a Run)
Given a run $\rho = q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n \in \mathit{Runs}(A)$, let $w_i = \boldsymbol{\pi}_{/\Sigma}(\mathit{tr}(\rho(i))), 0 \leq i \leq n$. The *cost* of $\rho \in \mathit{Runs}(A)$ is defined by:

$$\mathit{Cost}_2(\rho, A, \mathrm{Obs}) = \frac{1}{n+1} \cdot \sum_{i=0}^{n} |L(\delta(s_0, w_i))|.$$

■

We recall that $\mathit{Runs}^n(A)$ is the set of runs of length n in $\mathit{Runs}(A)$. The cost of the runs of length n of A is defined by:

$$\mathit{Cost}_2(n, A, \mathrm{Obs}) = \max\{\mathit{Cost}_2(\rho, A, \mathrm{Obs}) \text{ for } \rho \in \mathit{Runs}^n(A)\}.$$

And finally, the cost of the pair (Obs, A) is

$$\mathit{Cost}_2(A, \mathrm{Obs}) = \limsup_{n \to \infty} \mathit{Cost}_2(n, A, \mathrm{Obs}).$$

Notice that $\mathit{Cost}_2(n, A, \mathrm{Obs})$ is defined for each n because we have assumed A generates runs of arbitrary large length.

As emphasised previously, in order to compute $\mathit{Cost}_2(n, A, \mathrm{Obs})$ we consider that τ and f are now observable events, say u, but that the observer never chooses to observe them. Let $\mathrm{Obs}^+ = (S, s_0, \Sigma^u, \delta', L)$ where δ' is δ augmented with u-transitions that loop on each state $s \in S$. Let A^+ be A where τ and f transitions are renamed u. Let $A^+ \times \mathrm{Obs}^+$ be the synchronized product of A^+ and Obs^+. $A^+ \times \mathrm{Obs}^+ = (Z, z_0, \Sigma^u, \Delta)$ is complete w.r.t. Σ^u and we let $w(q, s) = |L(s)|$ so that $(A^+ \times \mathrm{Obs}^+, w)$ is a weighted automaton.

Theorem 6.1. $\mathit{Cost}_2(A, \mathrm{Obs}) = \nu^*(A^+ \times \mathrm{Obs}^+)$.

Proof:
The proof follows easily from the definitions. Let ρ be a run of A. There exists a run $\tilde{\rho}$ in $A^+ \times \mathrm{Obs}^+$ s.t. $\mathit{Cost}_2(\rho, A, \mathrm{Obs}) = \mu(\tilde{\rho})$ (μ is the mean cost as stated in Definition 6.3). $\tilde{\rho}$ is obtained from ρ by replacing τ and f transitions by some u transitions. Conversely for any run $\tilde{\rho}$ in $A^+ \times \mathrm{Obs}^+$ there is a run ρ in A s.t. $\mu(\tilde{\rho}) = \mathit{Cost}_2(\rho, A, \mathrm{Obs})$. □

We can compute the cost of a given pair (A, Obs): this can be done using Karp's maximum mean weight cycle algorithm [12] on weighted graphs. This algorithm is polynomial in the size of the weighted graph and thus we have:

Theorem 6.2. Computing the cost of (A, Obs) is in P.

Proof:
The size of $A^+ \times \text{Obs}^+$ is polynomial in the size of A and Obs. □

Remark 6.1. Notice that instead of the values $|L(s)|$ we could use any mapping from states of Obs to $\mathbb{Z}$ and consider that the cost of observing $\{a, b\}$ is less than observing a.

Example 6.1. We give the results for the computation of the cost of two observers for the DES $\mathcal{A}$ given in Fig. 2. Let O_1 be the most powerful observer that observes $\{a, b\}$ at each step, and O_2 be the observer given in Fig. 7.

The automata $\mathcal{A}^+ \times O_1^+$ and $\mathcal{A}^+ \times O_2^+$ are given in Fig. 13 and Fig. 14. The weight function is pictured above each state. Notice that to compute $\nu^*(A^+ \times O_i^+)$ we do not need the labels of the transitions as we are dealing with weighted graphs: if two transitions (s, a, s') and (s, b, s') are in $A^+ \times O_i^+$ we only need one of them. For instance in Fig. 13 one of the transitions $(0, a, 4)$ and $(0, b, 4)$ is redundant. The values $D_k(v)$ and $\min(v)$ for each state v of $A^+ \times O_i^+$ are given in Table 1 and Table 2. The maximum mean-weight value ν^* is the maximum value $\max_v \min(v)$ for v ranging over the set of states of $A^+ \times O_i^+$. We obtain $\mathit{Cost}_2(A, O_1) = 2$ and $\mathit{Cost}_2(A, O_2) = 1$.

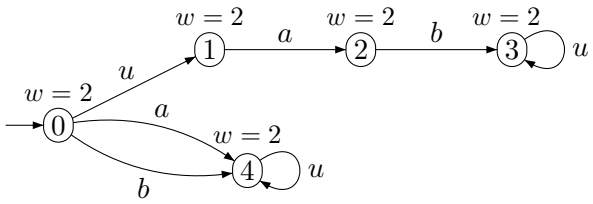


Figure 13. $\mathcal{A}^+ \times O_1^+$.

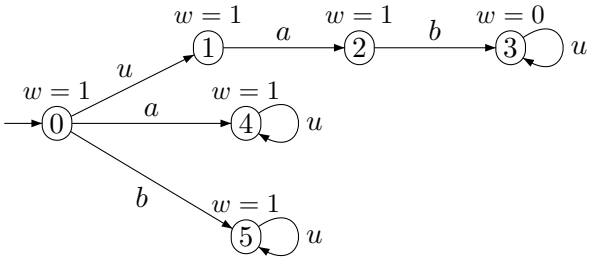


Figure 14. $\mathcal{A}^+ \times O_2^+$.

6.3. Optimal Dynamic Diagnosers

In this section, we focus on the problem of computing a best observer in the sense that diagnosing the DES with it has minimal cost. We address the following problem:

	0	1	2	3	4
D_0	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$
D_1	$-\infty$	4	$-\infty$	$-\infty$	4
D_2	$-\infty$	$-\infty$	6	$-\infty$	6
D_3	$-\infty$	$-\infty$	$-\infty$	8	8
D_4	$-\infty$	$-\infty$	$-\infty$	10	10
min	$-\infty$	$-\infty$	$-\infty$	**2**	**2**

Table 1. Iterations for $\mathcal{A}^+ \times O_1^+$.

	0	1	2	3	4	5
D_0	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
D_1	$-\infty$	2	$-\infty$	$-\infty$	2	2
D_2	$-\infty$	$-\infty$	3	$-\infty$	3	3
D_3	$-\infty$	$-\infty$	$-\infty$	4	4	4
D_4	$-\infty$	$-\infty$	$-\infty$	4	5	5
D_5	$-\infty$	$-\infty$	$-\infty$	4	6	6
min	$-\infty$	$-\infty$	$-\infty$	0	**1**	**1**

Table 2. Iterations for $\mathcal{A}^+ \times O_2^+$.

Problem 8. (Bounded Cost Observer)

INPUT: Plant model $A = (Q, q_0, \Sigma^{\tau,f}, \delta)$, $k \in \mathbb{N}$ and $c \in \mathbb{N}$.
PROBLEM:

(A). Is there an observer Obs s.t. A is (Obs,k)-diagnosable and $\mathit{Cost}_2(\text{Obs}) \leq c$?

(B). If the answer to (A) is "yes", compute a witness observer Obs, that satisfies $\mathit{Cost}_2(\text{Obs}) \leq c$.

Theorem 5.7, page 28 establishes that there is a most permissive observer $\mathcal{F}_A$ in case A is (Σ, k)-diagnosable and it can be computed in exponential time in the size of A and k, doubly exponential time in $|\Sigma|$, and has size exponential in A and k, and doubly exponential in $|\Sigma|$. Moreover the most permissive observer $\mathcal{F}_A$ can be represented by a finite state machine $S_{\mathcal{F}_A} = (\{0, 2 \cdots, l\} \cup (\{1, 3, \cdots, 2l'+1\} \times 2^\Sigma), 0, \Sigma \cup 2^\Sigma, \delta)$ which has the following properties:

- even states are states where the observer chooses a set of events to observe;
- odd states $(2i+1, X)$ are states where the observer waits for an observable event in X to occur;
- if $\delta(2i) = (2i'+1, X)$ with $X \in 2^\Sigma$, it means that from an even state $2i$, the automaton $S_{\mathcal{F}_A}$ can select a set X of events to observe. The successor state is an odd state together with the set X of events that are being observed;
- if $\delta((2i+1, X), a) = 2i'$ with $a \in X$, it means that from $(2i+1, X)$, $S_{\mathcal{F}_A}$ is waiting for an observable event to occur. When some occurs it switches to an even state.

By definition of $\mathcal{F}_A$, any observer O s.t. A is (O, k)-diagnosable must select a set of observable events in $\mathcal{F}_A(\mathit{tr}(\overline{w}))$ after having observed $w \in \boldsymbol{\pi}_{/\Sigma}(\mathcal{L}(A))$.

To compute an optimal observer, we use a result by Zwick and Paterson [24] on *weighted graph games*. The next subsection summarizes the results we are going to use.

6.3.1. Zwick and Paterson's Algorithm

Definition 6.4. (Weighted Graph)
A *weighted directed graph* is a pair (G, w) s.t. $G = (V, E)$ is a directed graph and $w : E \to \{-W, \cdots, 0, \cdots, W\}$ assigns an integral weight to each edge of G with $W \in \mathbb{N}$. We assume that each vertex $v \in V$ is reachable from a unique *source* vertex v_0 and has at least one outgoing transition. ■

Definition 6.5. (Weighted Graph Game)
A *weighted graph game* $G = (V, E)$ is a bipartite weighted graph with $V = V_1 \cup V_2$ and $E = E_1 \cup E_2$, $E_1 \subseteq V_1 \times V_2$ and $E_2 \subseteq V_2 \times V_1$. We assume the initial vertex v_0 of G belongs to V_1. ■

Vertices V_i are Player i's vertices. A weighted graph game is a turn based game in which the turn alternates between Player 1 and Player 2. The game starts at a vertex $v_0 \in V_1$. Player 1 chooses an edge $e_1 = (v_0, v_1)$ and then Player 2 chooses an edge $e_2 = (v_1, v_2)$ and so on and they build an infinite sequence of edges. Player 1 wants to maximise $\liminf_{n\to\infty} \frac{1}{n} \cdot \sum_{i=1}^{n} w(e_i)$ and Player 2 wants to minimize $\limsup_{n\to\infty} \frac{1}{n} \cdot \sum_{i=1}^{n} w(e_i)$.

One of the result of [24] is that there is a rational value $\nu \in \mathbb{Q}$ s.t. Player 1 has a strategy to ensure $\liminf_{n\to\infty} \frac{1}{n} \cdot \sum_{i=1}^{n} w(e_i) \geq \nu$ and Player 2 has a strategy to ensure that $\limsup_{n\to\infty} \frac{1}{n} \cdot \sum_{i=1}^{n} w(e_i) \leq \nu$. ν is called the value of the game. Let $n = |V|$. To compute ν, proceed as follows [24]:

1. Let $\nu_0(v) = 0$ for $v \in V$. For $v \in V$ and $k \geq 1$, $\nu_k(v)$ is defined by:

$$\nu_k(v) = \begin{cases} \max_{(v,w)\in E}\{w(v,w) + \nu_{k-1}(w)\} \text{ if } v \in V_1 \\ \min_{(v,w)\in E}\{w(v,w) + \nu_{k-1}(w)\} \text{ if } v \in V_2 \end{cases}$$

   This is the equivalent of the $D_k(v)$ values for Karp's algorithm using a min max strategy depending on which player is playing;

2. for each $v \in V$, compute $\nu'(v) = \nu_k(v)/k$ for $k = 4 \cdot n^3 \cdot W$.

3. for each vertex, the value of the game from v is the only rational number with a denominator at most n that lies in the interval $]\nu'(v) - \alpha, \nu'(v) + \alpha[$ with $\alpha = \frac{1}{2n(n-1)}$.

The value of the game is $\nu = \nu(v_0)$ where v_0 is the initial vertex. To compute an optimal strategy for Player 1, proceed as follows:

1. compute the values $\nu(v)$ for each $v \in V$;

2. if all the vertices of V_1 have outgoing degree 1, there is a unique strategy and it is positional and optimal;

3. otherwise, take a vertex $v \in V_1$ with outgoing degree $d \geq 2$. Remove $\lceil \frac{d}{2} \rceil$ edges from v leaving at least one. Recompute the value m_v for each v. If $m_v = \nu(v)$, there is an optimal positional strategy which uses the remaining edges from v. Otherwise there is a positional strategy that uses one of the removed edges.

We can iterate the previous scheme to find an optimal strategy for Player 1. In summary some of the results by Zwick and Paterson [24] we are going to use are:

- there is a value $\nu \in \mathbb{Q}$, called the *value of the game* s.t. Player 1 has a strategy to ensure that $\liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(e_i) \geq \nu$ and Player 2 has a strategy to ensure that $\limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(e_i) \leq \nu$; this value can be computed in $O(|V|^3 \times |E| \times W)$ where W is the range of the weight function (assuming the weights are in the interval $[-W..W]$). Note that deciding whether this value satisfies $\nu \bowtie c$ for $\bowtie \in \{=, <, >\}$ for $c \in \mathbb{Q}$ can be done in $O(|V|^2 \times |E| \times W)$.

- there are optimal memoryless strategies for both players that can be computed in $O(|V|^4 \times |E| \times \log(|E|/|V|) \times W)$.

6.3.2. Synthesis of an Optimal Observer

To solve the Problem 8, we use the most permissive observer $\mathcal{F}_A$ we computed in section 5.5. Given A and $\mathcal{F}_A$, we build a weighted graph game $G(A, \mathcal{F}_A)$ s.t. the value of the game is the optimal cost for the set of all observers. Moreover an optimal observer can be obtained by taking an optimal memoryless strategy in $G(A, \mathcal{F}_A)$.

To build $G(A, \mathcal{F}_A)$ we use the same idea as in section 6.2: we replace τ and f transitions in A by u obtaining A^+. We also modify $\mathcal{F}_A$ to obtain a weighted graph game $(\mathcal{F}_A^+, w)$ by adding transitions so that each state $2k+1$ is complete w.r.t. Σ^u. This is done as follows:

- from each $(2i+1, X)$ state, create a new even state i.e., pick some $2i'$ that has not already been used. Add transitions $((2i+1, X), \sigma, 2i')$ for each $\sigma \in \Sigma^u \setminus \mathit{Enabled}(2i+1, X)$. Add also a transition $(2i', X, (2i+1, X))$. This step means that if a A produces an event and it is not observable, $\mathcal{F}_A^+$ just reads the event and makes the same choice again.
- the weight of a transition $(2i, X, (2i'+1, X))$ is $|X|$.

The automaton $\mathcal{F}_A^+$ obtained from $\mathcal{F}_A$ is depicted on Fig. 15. The game $G(A, \mathcal{F}_A)$ is then $A^+ \times \mathcal{F}_A^+$. This

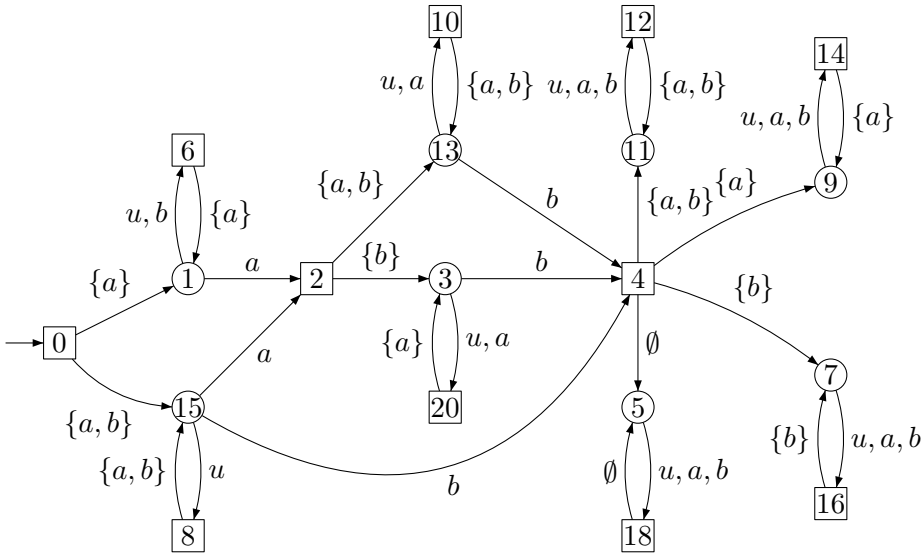


Figure 15. The Automaton $\mathcal{F}_A^+$.

way we can obtain a weighted graph game $WG(A, \mathcal{F}_A)$ by abstracting away the labels of the transitions. Notice that it still enables us to convert any strategy in $WG(A, \mathcal{F}_A)$ to a strategy in $\mathcal{F}_A$. A strategy in $WG(A, \mathcal{F}_A)$ will define an edge $(2i, (2i'+1, X))$ to take. As the target vertex contains the set of events we chose to observe we can define a corresponding strategy in $\mathcal{F}_A$.

By construction of $G(A, \mathcal{F}_A)$ and the definition of the value of a weighted graph game, the value of the game is the optimal cost for the set of all observers O s.t. A is (O, k)-diagnosable.

Assume A has n states and m transitions. From Theorem 5.7 we know that $\mathcal{F}_A$ has at most $O(2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$ states and $O(2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}} \times n^2 \times k \times m)$ transitions. Hence $G(A, \mathcal{F}_A)$ has at most $O(n \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$ vertices and $O(m \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$ edges. To make the game complete we may add at most half the number of states and hence $WG(A, \mathcal{F}_A)$ has the same size. We thus obtain the following results:

Theorem 6.3. Problem 8 can be solved in time $O(|\Sigma| \times m \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$.

We can even solve the optimal cost computation problem:

Problem 9. (Optimal Cost Observer)

INPUT: Plant model $A = (Q, q_0, \Sigma^{\tau,f}, \delta)$, $k \in \mathbb{N}$.
PROBLEM: Compute the least value m s.t. there exists an observer Obs s.t. A is (Obs,k)-diagnosable and $\mathit{Cost}_2(\text{Obs}) \leq m$.

Theorem 6.4. Problem 9 can be solved in time $O(|\Sigma| \times m \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$.

A consequence of Theorem 6.4 and Zwick and Paterson's results is that the cost of the optimal observer is a rational number.

Example 6.2. For the example $\mathcal{A}$ of Fig. 8 and $\mathcal{F}_A^+$ of Fig. 15, using Zwick and Paterson's algorithm we obtain that the optimal cost is 1 and the optimal strategy is to use the observer Obs of Fig. 7.

7. Conclusions

In this paper we have addressed sensor minimization problems in the context of fault diagnosis, using both static and dynamic observers. We showed that computing the smallest number of observable events necessary to achieve diagnosis with a static observer is NP-complete: this result also holds in the mask-based setting which allows to consider events that are observable but not distinguishable. We then focused on dynamic observers and proved that, for a given such observer, diagnosability can be checked in polynomial time (as in the case of static observers). We also solved a synthesis problem of dynamic observers and showed that a most-permissive dynamic observer can be computed in doubly-exponential time, provided an upper bound on the delay needed to detect a fault is given. Finally we have defined a notion of cost for dynamic observers and shown how to compute the minimal-cost observer that can be used to detect faults within a given delay.

There are several directions we are currently investigating:

- Problem 4 has not been solved so far. The major impediment to solve it is that the reduction we propose in section 5 yields a Büchi game. The algorithm we give in section 5.3 does not work for Büchi games and cannot be extended trivially. More generally we plan to extend the framework we have introduced for fault diagnosis to control under dynamic partial observation and this will enable us to solve Problem 4.

- Problem 5 is solved in doubly exponential time. To reduce the number of states of the most permissive observer, we point out that only *minimal* sets of events need to be observed. Indeed, if we can diagnose a system by observing only A from some point on, we surely can diagnose it using any superset $A' \supseteq A$. So far we keep all the sets that can be used to diagnose the system. We could possibly take advantage of the previous property using techniques described in [7].

Acknowledgments

We would like to thank Karine Altisen for her contribution during earlier phases of this work. We would also like to thank the anonymous reviewers for their careful reading, helpful comments and pointers to related work.

References

[1] Cassez, F., Tripakis, S., Altisen, K.: Sensor Minimization Problems with Static or Dynamic Observers for Fault Diagnosis, *7th Int. Conf. on Application of Concurrency to System Design (ACSD'07)*, IEEE Computer Society, 2007.

[2] Cassez, F., Tripakis, S., Altisen, K.: Synthesis Of Optimal Dynamic Observers for Fault Diagnosis of Discrete-Event Systems, *1st IEEE & IFIP Int. Symp. on Theoretical Aspects of Soft. Engineering (TASE'07)*, IEEE Computer Society, 2007.

[3] Cieslak, R., Desclaux, C., Fawaz, A., Varaiya, P.: Supervisory control of discrete-event processes with partial observations, *IEEE Transactions on Automatic Control*, **33**, 1988, 249–260.

[4] Courcoubetis, C., Vardi, M., Wolper, P., Yannakakis, M.: Memory Efficient Algorithms for the Verification of Temporal Properties, *Formal Methods in System Design*, **1**, 1992, 275–288.

[5] Dasdan, A., Irani, S., Gupta, R.: Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems, *Annual ACM IEEE Design Automation Conference*, ACM Press New York, NY, USA, New Orleans, Louisiana, United States, 1999, ISBN:1-58133-109-7.

[6] Debouk, R., Lafortune, S., Teneketzis, D.: On an Optimization Problem in Sensor Selection, *Discrete Event Dynamic Systems*, **4**(12), November 2004.

[7] Doyen, L., Chatterjee, K., Henzinger, T., Raskin, J.-F.: Algorithms for omega-regular games with imperfect information, in: *Computer Science Logic (CSL)*, Lecture Notes in Computer Science 4207, Springer, 2006, 287–302.

[8] Holzmann, G., Peled, D., Yannakakis, M.: On nested depth-first search, *Proc. of the 2nd Spin Workshop*, American Mathematical Society, 1996.

[9] Holzmann, G. J.: Software model checking with SPIN, *Advances in Computers*, **65**, 2005, 78–109.

[10] Jiang, S., Huang, Z., Chandra, V., Kumar, R.: A Polynomial Algorithm for Testing Diagnosability of Discrete Event Systems, *IEEE Transactions on Automatic Control*, **46**(8), August 2001.

[11] Jiang, S., Kumar, R., Garcia, H.: Optimal Sensor Selection for Discrete Event Systems with Partial Observation, *IEEE Transactions on Automatic Control*, **48**(3), March 2003, 369–381.

[12] Karp, R.: A characterization of the minimum mean cycle in a digraph, *Discrete Mathematics*, **23**, 1978, 309–311.

[13] Kupferman, O., Vardi, M.: Synthesis with Incomplete Informatio, *2nd International Conference on Temporal Logic*, Manchester, July 1997.

[14] Lamouchi, H., Thistle, J.: Effective Control Synthesis for DES Under Partial Observations, *IEEE Conference on Decision and Control*, 2000.

[15] Ramadge, P., Wonham, W.: Supervisory control of a class of discrete event processes, *SIAM J. Control Optim.*, **25**(1), January 1987.

[16] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of Discrete Event Systems, *IEEE Transactions on Automatic Control*, **40**(9), September 1995.

[17] Thomas, W.: On the Synthesis of Strategies in Infinite Games, *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, 900, Springer, 1995, Invited talk.

[18] Thorsley, D., Teneketzis, D.: Active Acquisition of Information for Diagnosis and Supervisory Control of Discrete Event Systems, *Discrete Event Dynamic Systems*, **4**(17), December 2007, 531–583.

[19] Tsitsiklis, J.: On the Control of Discrete Event Dynamical Systems, *Mathematics of Control, Signals and Systems*, **2**(2), 1989.

[20] Yoo, T.-S., Garcia, H.: Computation of Fault Detection Delay in Discrete-Event Systems, *14th International Workshop on Principles of Diagnosis, DX'03*, Washington, D.C., June 2003.

[21] Yoo, T.-S., Lafortune, S.: On the computational complexity of some problems arising in partially-observed discrete event systems, *American Control Conference (ACC'01)*, 2001, Arlington, VA.

[22] Yoo, T.-S., Lafortune, S.: NP-completeness of sensor selection problems arising in partially observed discrete-event systems, *IEEE Transactions on Automatic Control*, **47**(9), September 2002, 1495–1499.

[23] Yoo, T.-S., Lafortune, S.: Polynomial-Time Verification of Diagnosability of Partially-Observed Discrete-Event Systems, *IEEE Transactions on Automatic Control*, **47**(9), September 2002, 1491–1495.

[24] Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs, *Theoretical Computer Science*, **158**(1–2), 1996, 343–359.

A. Proof of Theorem 5.2

The following fact holds for G_H:

Fact A.1. By definition G_H is deterministic. Hence for any word $w \in \mathcal{L}(G)$, there is a unique run $\beta(w) = s_0 \xrightarrow{w} s'$ in G_H with $tr(\beta(w)) = w$ and a unique (last) state of G_H after reading w which is $\Delta(s_0, w) = s'$.

Notice that Fact A.1 holds because u transitions can only happen as the last transition of a run in G_H.

Only If Part Let f_H be a winning strategy in G_H (as the safety objective is fixed we omit it). Let $\rho \in \mathit{Runs}_1(G)$ and let $f(\rho) = f_H(\beta(tr(\rho)))$. f is trace-based by its definition and Fact A.1.

f is also winning. To prove this we use the following Lemma: Given ρ a run of $Out(G, f)$, we can build a run $\rho_H = S_0\sigma_0 \cdots S_{n_\rho}$ in G_H, such that ρ_H is the longest run that can match ρ. It is defined as follows:

- $S_0 = \{q_0^0\}$,
- for $1 \leq 2i < n_\rho$, $S_{2i} = \mathit{Next}_2(S_{2i-1}, \lambda_{2i-1})$
- for $1 \leq 2i+1 < n_\rho$, $S_{2i+1} = \mathit{Next}_1(S_{2i}, \sigma_{2i})$
- either $S_{n_\rho} \in \{\bot_1, \bot_2\}$ or
  - if $n_\rho = 2k$, $S_{n_\rho} = \mathit{Next}_2(S_{2k-1}, \lambda_{2k-1})$,
  - if $n_\rho = 2k+1$, $S_{n_\rho} = \mathit{Next}_1(S_{2k}, \sigma_{2k})$.

Lemma A.1. This run ρ_H is in $Out(G_H, f_H)$ and has the properties:

P_1: for $1 \leq 2i < n_\rho$, $q_{2i}^0 \in S_{2i}$ and for $1 \leq 2i+1 < n_\rho$, $q_{2i+1}^1 \in S_{2i+1}$;

P_2: either (a) $tgt(\rho_H) = S_{n_\rho} \in \{\bot_1, \bot_2\}$ or (b) if $n_\rho = 2k$, $q_{2k}^0 \in S_{n_\rho}$, and if $n_\rho = 2k+1$, $q_{2k+1}^1 \in S_{n_\rho}$.

Proof:
We prove properties Lemma A.1 by induction on the number of Player 1 moves in ρ. Assume $k = 0$. Clearly the run $\rho_H = \{q_0^0\}$ is in $Out(G_H, f_H)$ and Lemma A.1 holds.

Assume the property holds for k Player 1 moves. Let ρ be a run with $k+1$ Player 1 moves $\sigma_0, \sigma_2, \cdots, \sigma_{2k}$. We write

$$
\begin{array}{rcl}
\rho & = & q_0^0 \xrightarrow{\sigma_0} q_1^1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_1^{n_1} \xrightarrow{\lambda_1} \\
 & & q_2^0 \xrightarrow{\sigma_2} q_3^1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_3^{n_3} \xrightarrow{\lambda_1} \\
 & & \vdots \\
 & & q_{2k}^0 \xrightarrow{\sigma_{2k}} q_{2k+1}^1 \xrightarrow{\tau} \cdots q_{2k+1}^{n_{2k+1}} \xrightarrow{\lambda_{2k+1}} \\
 & & q_{2(k+1)}^0 \xrightarrow{\sigma_{2(k+1)}} q_{2(k+1)+1}^1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_{2(k+1)+1}^{n_{2(k+1)+1}}
\end{array}
$$

Let $\rho' = q_0^0 \cdots q_{2k+1}^{n_{2k+1}}$. ρ' has $k-1$ Player 1 moves and by induction hypothesis on ρ', we obtain:

- $\rho'_H \in Out(G_H, f_H)$ and P_1 and P_2 hold;
- as ρ'_H satisfies P_2:
  - either it satisfies P_2.(a) and $tgt(\rho'_H) \in \{\bot_1, \bot_2\}$; in this case ρ'_H cannot be extended in G_H and thus $\rho_H = \rho'_H$ and ρ_H satisfies P_1 and P_2.(a);
  - or P_2.(b) holds and ρ'_H does not contain any state in $\{\bot_1, \bot_2\}$; Then $\rho'_H = S_0 \cdots S_{2k}\sigma_{2k}S_{2k+1}$ satisfies P_1 and P_2.(b) and we have in particular $q^0_{2k} \in S_{2k}$ and $q^1_{2k+1} \in S_{2k+1}$. Because $q^1_{2k+1} \xrightarrow{\tau} \cdots q^{n_{2k+1}}_{2k+1} \xrightarrow{\lambda_{2k+1}} q^0_{2(k+1)}$, the set $Next_2(S_{2k+1}, \lambda_{2k+1})$ is not empty and thus $= S_0 \cdots S_{2k}\sigma_{2k}S_{2k+1}\lambda_{2k+1}S_{2(k+1)}$ is in $Out(G_H, f_H)$ and $S_{2(k+1)} \neq \bot_2$.
    $f_H(S_0 \cdots \sigma_{2k}S_{2k+1}\lambda_{2k+1}S_{2(k+1)}) = f(q^0_0 \cdots q^0_{2(k+1)})$ by definition of f and $S_0\sigma_0 \cdots \sigma_{2k}S_{2(k+1)} = \beta(tr(q^0_0\sigma_0 \cdots q^0_{2(k+1)}))$. Two cases arise:
    1. either $S_{2(k+1)} \xrightarrow{\sigma_{2(k+1)}} \bot_2$:
       in this case $\rho_H = S_0 \cdots S_{2k} \cdots \lambda_{2k+1}S_{2(k+1)}\sigma_{2(k+1)}\bot_2$ and $\rho_H \in Out(G_H, f_H)$ and satisfies P_1 and P_2.(a);
    2. or $S_{2(k+1)} \xrightarrow{\sigma_{2(k+1)}} S_{2(k+1)+1}$.
       This means that $S_{2(k+1)+1} \neq \bot_2$. In this case $q^1_{2(k+1)+1} \in S_{2(k+1)+1}$. Again two cases arise:
       (a) either there is some $2 \leq j \leq n_{2(k+1)+1}$ s.t. $q^j_{2(k+1)+1} \notin F$ and in this case $S_{2(k+1)+1} \xrightarrow{u} \bot_1$ is in Δ and $\rho_H = S_0 \cdots S_{2(k+1)}\sigma_{2(k+1)}S_{2(k+1)+1}u\bot_1$ is in $Out(G_H, f_H)$. ρ_H satisfies P_1 and P_2.(a);
       (b) or $q^j_{2(k+1)+1} \in F$ for $1 \leq j \leq n_{2(k+1)+1}$ and in this case $\rho_H = S_0 \cdots S_{2(k+1)}\sigma_{2(k+1)}S_{2(k+1)+1}$ is in $Out(G_H, f_H)$ and satisfies P_1 and P_2.(b).

This completes the proof of Lemma A.1. □

Now assume f is not winning, we can build a run $\rho \in Out(G, f)$ with $tgt(\rho) \notin F$. Applying Lemma A.1 we obtain: there is a run $\rho_H \in Out(G_H, f_H)$ that satisfies P_1 and P_2:

- either $tgt(\rho_H) \in \{\bot_1, \bot_2\}$ and in this case ρ_H is a losing run for Player 1 and thus f_H is not winning which contradicts the assumption;
- or ρ_H contains no state in $\{\bot_1, \bot_2\}$. Two cases arise depending on the parity of the length of ρ_H:
  - either $\rho_H = S_0 \cdots S_{2k}$. In this case $tgt(\rho) \in S_{2k}$ (P_2.(b)) and thus ρ_H ends in a losing state and f_H is not winning.
  - or $\rho_H = S_0 \cdots S_{2k+1}$ and again $tgt(\rho) \in S_{2k+1}$ and ρ_H is losing and f is not winning.

  This contradicts the assumption that f_H is winning.

This concludes the proof of the *Only If* part.

If Part.

Let w be a run of G_H ending in W_1-state. If w does not contain any u action, there is a run $\rho \in \mathit{Runs}_1(G)$ s.t. $\mathit{tr}(\rho) = \mathit{tr}(w)$ by construction of G_H (Definition 5.8). We let $f_H(w) = f(\rho)$. If $w = S_0 \cdots S_k u \bot_1$, we let $f_H(w) = f_H(S_0 \cdots S_k)$.

f_H is well defined as if ρ, ρ' are such that $\mathit{tr}(\rho) = \mathit{tr}(\rho')$ then $f(\rho) = f(\rho')$ because f is trace-based.

f_H is winning. To prove this we use the following lemma:

Lemma A.2. Let $w = S_0\sigma_0 S_k$ be a run in $Out(G_H, f_H)$. Then one of two following conditions hold:

D_1: $S_k \in \{\bot_1, \bot_2\}$ and there is a run $q_0^0 \cdots q$ in $Out(G, f)$ such that either (i) with $q \notin F$ or (ii) $f(q_0^0 \cdots q) = \sigma$ and $\sigma \notin \mathit{Enabled}(q)$;

D_2: $S_i \notin \{\bot_1, \bot_2\}$ for $0 \leq i \leq k$ and for any $q \in S_k$, there is a run $\rho = q_0^0 \cdots q$ in $Out(G, f)$ s.t. $\mathit{tr}(\rho) = \mathit{tr}(w)$.

Proof:

We prove Lemma A.2 by induction of the number of Player 1 moves in w. If w contains 0 moves clearly it holds.

Assume w is a run of $Out(G_H, f_H)$ that contains $k+1$ Player 1 moves. There are two possible forms for w:

- w ends in a W_2-state. $w = S_0\sigma_0 \cdots S_{2k}\sigma_{2k}S_{2k+1}$. All the states S_i, $0 \leq i \leq 2k$ must be different from $\{\bot_1, \bot_2\}$. Then either $S_{2k+1} = \bot_2$ or not.
  - if $S_{2k+1} = \bot_2$, there is some state $q \in S_{2k}$ s.t. $\sigma_{2k} \notin \mathit{Enabled}(q)$ by definition of G_H. We can apply the induction assumption on the run $w' = S_0\sigma_0 \cdots S_{2k}$ that contains k Player 1 moves. This leads: there is a run $\rho' = q_0^0 \cdots q$ in $Out(G, f)$ s.t. $\mathit{tr}(\rho') = \mathit{tr}(w')$ (because w' does not contain any $\bot_{1,2}$ state). Moreover $f(\rho') = f(w') = \sigma_{2k}$, $\sigma_{2k} \notin \mathit{Enabled}(q)$ and and thus condition $D_1.(ii)$ holds for w.
  - otherwise $S_{2k+1} \neq \bot_2$. By definition of G_H, this implies that $S_{2k+1} = \mathit{Next}_1(S_{2k}, \sigma_{2k})$. For any $q \in S_{2k+1}$ there is some $q' \in S_{2k}$ s.t. $q' \xrightarrow{\sigma_{2k}} q$ in G_H. We can again apply the induction assumption on the run $w' = S_0\sigma_0 \cdots S_{2k}$ that contains k Player 1 moves. This leads: there is a run $\rho' = q_0^0 \cdots q'$ in $Out(G, f)$ s.t. $\mathit{tr}(\rho') = \mathit{tr}(w')$ and $f(\rho') = f(w') = \sigma_{2k}$. Hence $\rho = \rho' \xrightarrow{\sigma_{2k}} q$ is a run of $Out(G, f)$ and $\mathit{tr}(\rho) = \mathit{tr}(w)$. w satisfies D_2.
- w ends in a W_1-state. Let $w = S_0\sigma_0 \cdots S_{2k}S_{2k+1}\lambda_{2k+1}S_{2(k+1)}$. It must be the case that $S_{2k+1} \neq \bot_2$. Two cases arise:
  1. $\lambda_{2k+1} \neq u$. By definition of G_H, $S_{2(k+1)} = \mathit{Next}_2(S_{2k+1}, \lambda_{2k+1})$. Let $q \in S_{2(k+1)}$. By definition of Next_2, there is a state $q' \in S_{2k+1}$ s.t. $q' \xrightarrow{\tau} \cdots \xrightarrow{\lambda_{2k+1}} q$. As in the previous case for any $q' \in S_{2k+1}$ we can build a run $\rho' = q_0^0 \cdots q'$ in $Out(G, f)$ with $\mathit{tr}(\rho') = \mathit{tr}(S_0 \cdots S_{2k+1})$. Hence $\rho = \rho' \xrightarrow{\lambda_{2k+1}} q$ is a run of $Out(G, f)$ and satisfies D_2.
  2. $\lambda_{2k+1} = u$. In this case $S_{2(k+1)} = \bot_1$. This mean that there is a state $q' \in S_{2k+1}$ s.t. $q' \xrightarrow{\tau} \cdots \xrightarrow{\tau} q$ and $q \notin F$. Again we can build a run $\rho' = q_0^0 \cdots q'$ in $Out(G, f)$ with $\mathit{tr}(\rho') = \mathit{tr}(S_0 \cdots S_{2k+1})$ and clearly $\rho = \rho' \xrightarrow{\tau} \cdots \xrightarrow{\tau} q$ is run of $Out(G, f)$ and thus w satisfies $D_1.(i)$.

This completes the proof of Lemma A.2. □

Now assume f_H is not winning. There is a run $S_0 \cdots S_k$ in $Out(G_H, f_H)$ such that either (i) $S_k \in \{\bot_1, \bot_2\}$ or (ii) there is some $q \in S_k$ such that $q \notin F$. Applying Lemma A.2 we obtain:

- if (i) holds, D_1 holds and there is a run $\rho \in Out(G, f)$ s.t. either (a) $\mathit{tgt}(\rho) \notin F$ or (b) $f(\rho) \notin \mathit{Enabled}(\mathit{tgt}(\rho))$. If (a) holds f is not a winning strategy. If (b) holds f is not a strategy. In any case this contradicts the fact that f is a winning strategy.
- if (ii) holds, as S_k contains a state $q \notin F$, there is a run $q_0^0 \cdots q$ in $Out(G, f)$ and $q \notin F$ which again contradicts the fact that f is winning.

This completes the proof of Theorem 5.2.